

# Package ‘doRedis’

March 11, 2022

**Type** Package

**Title** 'Foreach' Parallel Adapter Using the 'Redis' Database

**Version** 3.0.1

**Date** 2022-3-9

**Author** B. W. Lewis <blewis@illposed.net>

**Maintainer** B. W. Lewis <blewis@illposed.net>

**Description** Create and manage fault-tolerant task queues for the 'foreach' package using the 'Redis' key/value database.

**BugReports** <https://github.com/bwlewis/doRedis/issues>

**Depends** R (>= 3.0), foreach(>= 1.3.0), iterators(>= 1.0.0), utils

**Imports** redux, stats

**License** GPL-2

**LazyLoad** yes

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-03-11 14:30:02 UTC

## R topics documented:

doRedis-package . . . . .	2
.doRedis . . . . .	2
jobs . . . . .	3
logger . . . . .	3
redisConnect . . . . .	4
redisDelete . . . . .	4
redisGet . . . . .	5
redisSet . . . . .	5
redisWorker . . . . .	6
registerDoRedis . . . . .	7

removeJob . . . . .	9
removeQueue . . . . .	9
setChunkSize . . . . .	10
setExport . . . . .	11
setFtinterval . . . . .	12
setPackages . . . . .	12
setProgress . . . . .	13
startLocalWorkers . . . . .	13
tasks . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

doRedis-package	<i>A Redis parallel back end for foreach.</i>
-----------------	---

---

### Description

The doRedis package implements an elastic parallel back end for foreach using the Redis key/value database.

### See Also

[registerDoRedis](#), [startLocalWorkers](#)

---

.doRedis	<i>internal function called by foreach</i>
----------	--

---

### Description

internal function called by foreach

### Usage

```
.doRedis(obj, expr, envir, data)
```

### Arguments

obj	a foreach object
expr	the expression to evaluate
envir	the expression environment
data	a list of parameters from registerDoRedis

### Value

the foreach result

---

jobs	<i>List doRedis jobs</i>
------	--------------------------

---

**Description**

List doRedis jobs

**Usage**

```
jobs(queue = "*")
```

**Arguments**

queue            List jobs for the specified queue, or set to "\*" to list jobs for all queues

**Value**

a data frame listing jobs by row with variables queue, id, user, host and time (submitted).

---

logger	<i>Print a timestamped message to the standard error stream.</i>
--------	--

---

**Description**

Use to help debug remote doRedis workers.

**Usage**

```
logger(msg)
```

**Arguments**

msg            a character message to print to the standard error stream

**Value**

The character message that was printed, decorated with time and system info.

---

redisConnect	<i>Explicitly connect to a Redis server.</i>
--------------	--

---

**Description**

This function is normally not needed, use the `redux` package functions instead, or simply `registerDoRedis`.

**Usage**

```
redisConnect(host = "localhost", port = 6379L, password, ...)
```

**Arguments**

host	character Redis host name
port	integer Redis port number
password	optional character Redis password
...	optional additional arguments for compatibility with old <code>redis</code> , ignored

**See Also**

[registerDoRedis](#), [redisWorker](#), [startLocalWorkers](#)

---

redisDelete	<i>A convenience function to delete a Redis key</i>
-------------	---

---

**Description**

A convenience function to delete a Redis key

**Usage**

```
redisDelete(key)
```

**Arguments**

key	(character or raw) Redis key name to delete
-----	---

**Value**

Redis status message

**See Also**

[hiredis](#)

---

redisGet	<i>A convenience function to return an R value from a Redis key.</i>
----------	--

---

**Description**

This function assumes the value associated with the Redis key is a serialized (binary) R value and unserializes it on return.

**Usage**

```
redisGet(key)
```

**Arguments**

key	(character or raw) Redis key name
-----	-----------------------------------

**Value**

Unserialized R value.

**See Also**

[hiredis](#)

---

redisSet	<i>A convenience function to set an R value in a Redis key</i>
----------	--

---

**Description**

This function serializes the val argument.

**Usage**

```
redisSet(key, val)
```

**Arguments**

key	(character or raw) Redis key name
val	R value to set

**Value**

Redis status message

**See Also**

[hiredis](#)

---

redisWorker                      *Initialize a doRedis worker process.*

---

### Description

The `redisWorker` function enrolls the current R session in one or more `doRedis` worker pools specified by the work queue names. The worker loop takes over the R session until the work queue(s) are deleted, after which the worker loop exits after the `linger` period, or until the worker has processed `iter` tasks. Running workers also terminate after network activity with Redis remains inactive for longer than the `timeout` period set in the `redisConnect` function. That value defaults internally to 30 seconds in `redisWorker`. You can increase it by including a `timeout=n` argument value.

### Usage

```
redisWorker(
  queue,
  host = "localhost",
  port = 6379,
  iter = Inf,
  linger = 30,
  log = stderr(),
  connected = FALSE,
  password = NULL,
  loglevel = 0,
  timelimit = 0,
  ...
)
```

### Arguments

<code>queue</code>	work queue name or a vector of queue names
<code>host</code>	Redis database host name or IP address
<code>port</code>	Redis database port number
<code>iter</code>	maximum number of tasks to process before exiting the worker loop
<code>linger</code>	timeout in seconds after which the work queue is deleted that the worker terminates
<code>log</code>	print messages to the specified file connection
<code>connected</code>	set to TRUE to reuse an existing open connection to Redis, otherwise establish a new one
<code>password</code>	optional Redis database password
<code>loglevel</code>	set to > 0 to increase verbosity in the log
<code>timelimit</code>	set to > 0 to specify a task time limit in seconds, after which worker processes are killed; beware that setting this value > 0 will terminate any R worker process if their task takes too long.
<code>...</code>	Optional additional parameters passed to <a href="#">redisConnect</a>

**Value**

NULL is invisibly returned.

**Note**

The worker connection to Redis uses a TCP timeout value of 30 seconds by default. That means that the worker will exit after about 30 seconds of inactivity. If you want the worker to remain active for longer periods, set the `timeout` option to a larger value.

Use the `linger` option to instruct the worker to linger for up to the indicated number of seconds after the listening work queue has been removed. After at most that interval, the worker will exit after removing the queue.

**See Also**

[registerDoRedis](#), [startLocalWorkers](#)

---

<code>registerDoRedis</code>	<i>Register the Redis back end for foreach.</i>
------------------------------	---

---

**Description**

The `doRedis` package implements a simple but flexible parallel back end for `foreach` that uses Redis for inter-process communication. The work queue name specifies the base name of a small set of Redis keys that the coordinator and worker processes use to exchange data.

**Usage**

```
registerDoRedis(  
  queue,  
  host = "localhost",  
  port = 6379,  
  password,  
  ftinterval = 30,  
  chunkSize = 1,  
  progress = FALSE,  
  ...  
)
```

**Arguments**

<code>queue</code>	A work queue name
<code>host</code>	The Redis server host name or IP address
<code>port</code>	The Redis server port number
<code>password</code>	An optional Redis database password
<code>ftinterval</code>	Default fault tolerance interval in seconds

chunkSize	Default iteration granularity, see <a href="#">setChunkSize</a>
progress	(logical) Show progress bar for computations?
...	Optional arguments passed to <a href="#">redisConnect</a>

### Details

Back-end worker R processes advertise their availability for work with the [redisWorker](#) function. The doRedis parallel back end tolerates faults among the worker processes and automatically re-submits failed tasks. It is also portable and supports heterogeneous sets of workers, even across operative systems. The back end supports dynamic pools of worker processes. New workers may be added to work queues at any time and can be used by running foreach computations.

### Value

NULL is invisibly returned; this function is called for side effect of registering a foreach backend.

### Note

All doRedis functions require access to a Redis database server (not included with this package).

Worker processes default to same random number generator as the coordinator process by default with seeds set per iteration rather than per worker to yield reproducible output independent of the number of worker processes. The L'Ecuyer-CMRG RNG available from the parallel package is recommended when high-quality distributed pseudorandom numbers are needed. See package vignette for more details and additional options.

Avoid using fork-based parallel functions within doRedis expressions. Use of `mclapply` and similar functions in the body of a doRedis foreach loop can result in worker faults.

### See Also

[foreach](#), [doRedis-package](#), [setChunkSize](#), [removeQueue](#)

### Examples

```
# Only run if a Redis server is running
if (redux::redis_available()) {
## The example assumes that a Redis server is running on the local host
## and standard port.

# 1. Start a single local R worker process
startLocalWorkers(n=1, queue="jobs", linger=1)

# 2. Run a simple sampling approximation of pi:
registerDoRedis("jobs")
pie = foreach(j=1:10, .combine=sum, .multicombine=TRUE) %dopar%
      4 * sum((runif(1000000) ^ 2 + runif(1000000) ^ 2) < 1) / 1000000
removeQueue("jobs")
print(pie)

# Note that removing the work queue automatically terminates worker processes.
```

```
}

```

---

removeJob	<i>Remove Redis keys associated with one or more doRedis jobs</i>
-----------	---

---

### Description

Remove Redis keys associated with one or more doRedis jobs

### Usage

```
removeJob(job)
```

### Arguments

job	Either a named character vector with "queue" and "id" entries corresponding to a doRedis job queue and job id, or a list with equal-length "queue" and "id" entries, or a data frame with "queue" and "id" entries, for example as returned by <a href="#">jobs</a> .
-----	---

### Value

NULL is invisibly returned; this function is used for its side effect—in particular, removing all Redis keys associated with the specified job.

---

removeQueue	<i>Remove a doRedis queue and delete all associated keys from Redis.</i>
-------------	--

---

### Description

Removing a doRedis queue cleans up associated keys in the Redis database and signals to workers listening on the queue to terminate. Workers terminate after their timeout period after their work queue is deleted.

### Usage

```
removeQueue(queue)
```

### Arguments

queue	the doRedis queue name
-------	------------------------

### Value

NULL is invisibly returned; this function is called for the side effect of removing Redis keys associated with the specified queue.

**Note**

Workers listening for work on more than one queue will only terminate after all their queues have been deleted. See [registerDoRedis](#) for an example.

---

setChunkSize	<i>Set the default granularity of distributed tasks.</i>
--------------	--

---

**Description**

A job is the collection of all tasks in a foreach loop. A task is a collection of loop iterations of at most size chunkSize. R workers are assigned work by task in blocks of at most chunkSize loop iterations per task. The default value is one iteration per task. Setting the default chunk size larger for shorter-running jobs can substantially improve performance. Setting this value too high can negatively impact load-balancing across workers, however.

**Usage**

```
setChunkSize(value = 1)
```

**Arguments**

value            positive integer chunk size setting

**Value**

value is invisibly returned; this value is called for its side effect.

**Examples**

```
# Only run if a Redis server is running
if (redux::redis_available()) {

# Start a single local R worker process
startLocalWorkers(n=1, queue="jobs", linger=1)

# Register the work queue with the coordinator R process
registerDoRedis("jobs")

# Compare verbose task submission output from...
setChunkSize(1)
foreach(j=1:4, .combine=c, .verbose=TRUE) %dopar% j

# with the verbose task submission output from:
setChunkSize(2)
foreach(j=1:4, .combine=c, .verbose=TRUE) %dopar% j

# Clean up
removeQueue("jobs")
}
```

---

setExport	<i>Manually add symbol names to the worker environment export list.</i>
-----------	---

---

### Description

The setExport function lets users manually declare symbol names of corresponding objects that should be exported to workers.

### Usage

```
setExport(names = c())
```

### Arguments

names            A character vector of symbol names to export.

### Details

The foreach function includes a similar .export parameter.

We provide this supplemental export option for users without direct access to the foreach function, for example, when foreach is used inside another package.

### Value

The value of names is invisibly returned (this function is used ofr its side effect).

### Examples

```
## Not run:
registerDoRedis("work queue")
startLocalWorkers(n=1, queue="work queue", linger=1)

f <- function() pi

(foreach(1) %dopar% tryCatch(eval(call("f")), error = as.character))
# Returns the error converted to a message:
# Error in eval(call("f")) : task 1 failed - could not find function "f"

# Manually export the symbol f:
setExport("f")
(foreach(1) %dopar% eval(call("f")))
# Now f is found.

removeQueue("work queue")

## End(Not run)
```

---

setFtinterval	<i>Set the fault tolerance check interval in seconds.</i>
---------------	---

---

**Description**

Failed tasks are automatically re-submitted to the work queue. The `setFtinterval` sets an upper bound on how frequently the system checks for failure. See the package vignette for discussion and examples.

**Usage**

```
setFtinterval(value = 30)
```

**Arguments**

`value` positive integer number of seconds

**Value**

`value` is invisibly returned (this function is used for its side effect).

---

setPackages	<i>Manually set package names in the worker environment package list.</i>
-------------	---

---

**Description**

The `setPackages` function lets users manually declare packages that R worker processes need to load before running their tasks.

**Usage**

```
setPackages(packages = c())
```

**Arguments**

`packages` A character vector of package names.

**Details**

The `foreach` function includes a similar `.packages` parameter.

Defines a way to set the `foreach .packages` option for users without direct access to the `foreach` function, for example, when `foreach` is used inside another package.

**Value**

The value of `packages` is invisibly returned (this function is used for its side effect).

---

setProgress	<i>Progress bar</i>
-------------	---------------------

---

**Description**

Progress bar

**Usage**

```
setProgress(value = FALSE)
```

**Arguments**

value            if TRUE, display a text progress bar indicating status of the computation

**Value**

value is invisibly returned (this function is used for its side effect).

---

startLocalWorkers	<i>Start one or more background R worker processes on the local system.</i>
-------------------	---

---

**Description**

Use startLocalWorkers to start one or more doRedis R worker processes in the background. The worker processes are started on the local system using the redisWorker function.

**Usage**

```
startLocalWorkers(  
  n,  
  queue,  
  host = "localhost",  
  port = 6379,  
  iter = Inf,  
  linger = 30,  
  log = stdout(),  
  Rbin = paste(R.home(component = "bin"), "R", sep = "/"),  
  password,  
  ...  
)
```

**Arguments**

n	number of workers to start
queue	work queue name
host	Redis database host name or IP address
port	Redis database port number
iter	maximum number of tasks to process before exiting the worker loop
linger	timeout in seconds after which the work queue is deleted that the worker terminates
log	print messages to the specified file connection
Rbin	full path to the command-line R program
password	optional Redis database password
...	optional additional parameters passed to the <a href="#">redisWorker</a> function

**Details**

Running workers self-terminate after a `linger` period if their work queues are deleted with the `removeQueue` function, or when network activity with Redis remains inactive for longer than the timeout period set in the `redisConnect` function. That value defaults internally to 3600 (one hour) in `startLocalWorkers`. You can increase it by including a `timeout=n` argument value.

**Value**

NULL is invisibly returned.

**See Also**

[registerDoRedis](#), [redisWorker](#)

**Examples**

```
# Only run if a Redis server is running
if (redux::redis_available()) {
  ## The example assumes that a Redis server is running on the local host
  ## and standard port.

  # Start a single local R worker process
  startLocalWorkers(n=1, queue="R jobs", linger=1)

  # Run a simple sampling approximation of pi:
  registerDoRedis("R jobs")
  print(foreach(j=1:10, .combine=sum, .multicombine=TRUE) %dopar%
        4 * sum((runif(1000000) ^ 2 + runif(1000000) ^ 2) < 1) / 1000000)

  # Clean up
  removeQueue("R jobs")
}
```

---

tasks	<i>List running doRedis tasks</i>
-------	-----------------------------------

---

**Description**

List running doRedis tasks

**Usage**

```
tasks(queue = "*", id = "*")
```

**Arguments**

queue	List jobs for the specified queue, or set to "*" to list jobs for all queues
id	List tasks for the specified job id, or set to "*" to list tasks for all job ids

**Value**

a data frame listing jobs by row with variables queue, id, user, coordinator, time, iter, host, pid (see Note)

**Note**

The returned values indicate

1. queue the doRedis queue name
2. id the doRedis job id
3. user the user running the job
4. coordinator the host name or I.P. address where the job was submitted (and the coordinator R process runs)
5. time system time on the worker node when the task was started
6. iter the loop iterations being run by the task
7. host the host name or I.P. address where the task is running
8. pid the process ID of the R worker running the task on host

Tasks are listed until a key associated with them expires in Redis. Thus running tasks are not explicitly removed from the task list immediately when they terminate, but may linger on the list for a short while after (a few seconds).

# Index

.doRedis, [2](#)  
doRedis-package, [2](#)  
foreach, [8](#)  
hiredis, [4](#), [5](#)  
jobs, [3](#), [9](#)  
logger, [3](#)  
redisConnect, [4](#), [6](#), [8](#)  
redisDelete, [4](#)  
redisGet, [5](#)  
redisSet, [5](#)  
redisWorker, [4](#), [6](#), [8](#), [14](#)  
registerDoRedis, [2](#), [4](#), [7](#), [7](#), [10](#), [14](#)  
removeJob, [9](#)  
removeQueue, [8](#), [9](#)  
setChunkSize, [8](#), [10](#)  
setExport, [11](#)  
setFtinterval, [12](#)  
setPackages, [12](#)  
setProgress, [13](#)  
startLocalWorkers, [2](#), [4](#), [7](#), [13](#)  
tasks, [15](#)