# Package: future.mapreduce (via r-universe)

November 22, 2026

**Version** 0.0.1-9004

**Title** Utility Functions for Future Map-Reduce API Packages

**Depends** R (>= 3.2.0)

**Imports** future (>= 1.30.0), globals (>= 0.16.2), parallel, utils

**Description** Utility functions for other packages implementing map-reduce APIs on top of the future framework - specifically 'future.apply', 'furrr', and 'doFuture'. This package and its functions are intended for those packages only and should not be used elsewhere.

**License** GPL (>= 2)

**LazyLoad** TRUE

**URL** https://github.com/HenrikBengtsson/future.mapreduce

**BugReports** https://github.com/HenrikBengtsson/future.mapreduce/issues

**RoxygenNote** 7.2.3

**Roxygen** list(markdown = TRUE)

**Repository** https://futureverse-devel.r-universe.dev

**Date/Publication** 2026-09-24 02:35:31 UTC

**RemoteUrl** https://github.com/futureverse/future.mapreduce

**RemoteRef** develop

**RemoteSha** c6c405b97abeb20870aa48f6a33c3bb771697f27

## Contents


get_globals_and_packages_xapply . . . . . 2
make_chunks . . . . . 3
make_rng_seeds . . . . . 4

**Index** **5**

`get_globals_and_packages_xapply` *Identify Globals and Packages of a Map-Reduce Function Call*

## Description

Identify Globals and Packages of a Map-Reduce Function Call

## Usage

```
get_globals_and_packages_xapply(
  fun,
  fun_name = as.character(substitute(fun)),
  args = NULL,
  args_name = "...",
  globals = TRUE,
  packages = NULL,
  envir = parent.frame()
)
```

## Arguments

| | |
|---|---|
| `fun` | A function that takes one or more arguments. |
| `fun_name` | The name of the argument that `fun` should be passed as. |
| `args` | (optional) A list of arguments passed to `fun`, either via a named argument (`args_name`), or via .... |
| `args_name` | If `"..."`, then the arguments in `args` are passed to `fun()` as individual arguments. If a string, then `args` as passed to `fun()` via the argument of this name. |
| `globals` | (optional) a logical, a character vector, a named list, or a Globals object. If TRUE, globals are identified by code inspection based on `expr` and `tweak` searching from environment `envir`. If FALSE, no globals are used. If a character vector, then globals are identified by lookup based their names `globals` searching from environment `envir`. If a named list or a Globals object, the globals are used as is. |
| `packages` | (optional) a character vector specifying packages to be attached in the R environment evaluating the future. |
| `envir` | The environment from where globals should be searched. |

## Value

A names list with elements:

- `globals` - a FutureGlobals object
- `packages` - a character vector of package names

make_chunks *Create Chunks of Index Vectors*

## Description

*This is an internal function.*

## Usage

```
make_chunks(nelements, nworkers, scheduling = 1, chunk_size = NULL)
```

## Arguments

| | |
|---|---|
| nelements | (integer) Total number of elements to iterate over. |
| nworkers | (integer) Number of workers available. |
| scheduling | (numeric) A strictly positive scalar. Only used if argument chunk_size is NULL. |
| chunk_size | (numeric) The maximum number of elements per chunk, or NULL. If NULL, then the chunk sizes are given by the scheduling argument. |

## Value

A list of chunks, where each chunk is an integer vector of unique indices in [1, nelements]. The union of all chunks holds nelements elements and equals 1:nelements. If nelements == 0, then an empty list is returned.

## Control processing order of elements

Attribute ordering of chunk_size or scheduling can be used to control the ordering the elements are iterated over, which only affects the processing order *not* the order values are returned. This attribute can take the following values:

- index vector - an numeric vector of length nelements specifying how elements are remapped
- function - an function taking one argument which is called as ordering(nelements) and which must return an index vector of length nelements, e.g. function(n) rev(seq_len(n)) for reverse ordering.
- "random" - this will randomize the ordering via random index vector sample.int(nelements).

make_rng_seeds *Produce Reproducible Seeds for Parallel Random Number Generation*

## Description

Produce Reproducible Seeds for Parallel Random Number Generation

## Usage

```
make_rng_seeds(count, seed = FALSE)
```

## Arguments

| | |
|---|---|
| count | The number of RNG seeds to produce. |
| seed | A logical specifying whether RNG seeds should be generated or not. (seed = NULL corresponds to seed = FALSE). If a list, then it should be of length count and each element should consist of a valid RNG seed. |

## Details

This function generates count independent L'Ecuyer-CMRG random seeds that can be used as .Random.seed for parallel processing. These seeds are produced with help of parallel::nextRNGSubStream() and parallel::nextRNGStream() using a strategy that

```
seed <- <initial RNG seed>
for (ii in seq_len(count)) {
  seeds[[ii]] <- parallel::nextRNGSubStream(seed)
  seed <- parallel::nextRNGStream(seed)
}
```

This function forwards the RNG state 1 + count times if seed = TRUE.

## Value

Returns a non-named list of count independent L'Ecuyer-CMRG random seeds. If seed is NULL or FALSE, then NULL is returned.

## Examples

```
## Set up L'Ecuyer-CMRG random seeds for 100 parallel tasks
## based on .Random.seed in the current R process
seeds <- make_rng_seeds(100L, seed = TRUE)
str(seeds)
```

# Index

environment, *2*

function, *2*
FutureGlobals, *2*

get_globals_and_packages_xapply, 2
Globals, *2*

make_chunks, 3
make_rng_seeds, 4

parallel::nextRNGStream(), *4*
parallel::nextRNGSubStream(), *4*