Package: doRNG (via r-universe)

June 7, 2026

Type Package

Title Generic Reproducible Parallel Backend for 'foreach' Loops

Version 1.8.6.3

Date 2026-02-05

Encoding UTF-8

Description Provides functions to perform reproducible parallel foreach loops, using independent random streams as generated by L'Ecuyer's combined multiple-recursive generator [L'Ecuyer (1999), <DOI:10.1287/opre.47.1.159>]. It enables to easily convert standard '%dopar%' loops into fully reproducible loops, independently of the number of workers, the task scheduling strategy, or the chosen parallel environment and associated foreach backend.

License GPL (>= 2)

LazyLoad yes

URL https://github.com/emilioluissaenzguillen/doRNG

BugReports https://github.com/emilioluissaenzguillen/doRNG/issues

Depends R (>= 3.0.0), foreach, rngtools (>= 1.5)

Imports stats, utils, iterators

Suggests doParallel, doMPI, doRedis, rbenchmark, knitr, rbibutils (>= 1.3), testthat, covr

RoxygenNote 7.3.3

NeedsCompilation no

Repository https://emilioluissaenzguillen.r-universe.dev

Date/Publication 2026-02-07 14:39:46 UTC

RemoteUrl https://github.com/emilioluissaenzguillen/dorng

RemoteRef HEAD

RemoteSha 894684353d77d6e71485385aecb102c15b04d031

Contents

---

doRNG-package *Generic Reproducible Parallel Backend for foreach Loops*

Description

The *doRNG* package provides functions to perform reproducible parallel foreach loops, using independent random streams as generated by L'Ecuyer's combined multiple-recursive generator [L'Ecuyer (1999)]. It enables to easily convert standard independently of the number of workers, the task scheduling strategy, or the chosen parallel environment and associated foreach backend. It has been tested with the following foreach backend: doMC, doSNOW, doMPI.

References

L'Ecuyer, P. (1999). Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators. *Operations Research*, 47(1), 159–164. doi:10.1287/opre.47.1.159

See Also

doRNG, RNGseq

Examples

```
# Register a parallel backend (suggested package)
local({
  if (requireNamespace("doParallel", quietly = TRUE)) {
    cl <- parallel::makeCluster(2)
    doParallel::registerDoParallel(cl)
    on.exit(parallel::stopCluster(cl), add = TRUE)

    ## standard %dopar% loops are not reproducible
    set.seed(123)
    r1 <- foreach(i = 1:4) %dopar% { runif(1) }
    set.seed(123)
    r2 <- foreach(i = 1:4) %dopar% { runif(1) }
    identical(r1, r2)


    ## %dorng% loops _are_ reproducible
    set.seed(123)
    d1 <- foreach(i = 1:4) %dorng% { runif(1) }
    set.seed(123)
```

```
d2 <- foreach(i = 1:4) %dorng% { runif(1) }
identical(d1, d2)


# alternative way of seeding
a1 <- foreach(i = 1:4, .options.RNG = 123) %dorng% { runif(1) }
a2 <- foreach(i = 1:4, .options.RNG = 123) %dorng% { runif(1) }
identical(a1, a2) && identical(a1, d1)


## sequences of %dorng% loops _are_ reproducible
set.seed(123)
s1 <- foreach(i = 1:4) %dorng% { runif(1) }
s2 <- foreach(i = 1:4) %dorng% { runif(1) }
identical(s1, d1) && !identical(s1, s2)


set.seed(123)
s1.2 <- foreach(i = 1:4) %dorng% { runif(1) }
s2.2 <- foreach(i = 1:4) %dorng% { runif(1) }
identical(s1, s1.2) && identical(s2, s2.2)


## Non-invasive way of converting %dopar% loops into reproducible loops
registerDoRNG(123)
s3 <- foreach(i = 1:4) %dopar% { runif(1) }
s4 <- foreach(i = 1:4) %dopar% { runif(1) }
identical(s3, s1) && identical(s4, s2)


  } else {
    message("Package 'doParallel' is not installed; skipping parallel examples.")
  }
})
```

`%dorng%` *Reproducible Parallel Foreach Backend*

Description

‘ ‘

Usage

```
obj %dorng% ex
```

Arguments

obj	a foreach object as returned by a call to foreach.
ex	the R expression to evaluate.

Value

‘ The whole sequence of RNG seeds is stored in the result object as an attribute. Use `attr(res, 'rng')` to retrieve it.

Global options

These options are for advanced users that develop ‘foreach backends:

* ’doRNG.rng_change_warning_skip’: if set to a single logical ‘FALSE/TRUE‘, it indicates whether a warning should be thrown if the RNG seed is changed by the registered parallel backend (default=FALSE). Set it to ‘TRUE‘ if you know that running your backend will change the RNG state and want to disable the warning. This option can also be set to a character vector that specifies the name(s) of the backend(s) for which the warning should be skipped.

See Also

foreach, doParallel, registerDoParallel, doMPI

Examples

```
if (requireNamespace("doParallel", quietly = TRUE)) {
  cl <- parallel::makeCluster(2)
  doParallel::registerDoParallel(cl)
  on.exit(parallel::stopCluster(cl), add = TRUE)

  # standard %dopar% loops are _not_ reproducible
  set.seed(1234)
  s1 <- foreach(i=1:4) %dopar% { runif(1) }
  set.seed(1234)
  s2 <- foreach(i=1:4) %dopar% { runif(1) }
  identical(s1, s2)

  # single %dorng% loops are reproducible
  r1 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
  r2 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
  identical(r1, r2)
  # the sequence of RNG seeds is stored as an attribute
  attr(r1, "rng")
} else {
  message("Package 'doParallel' is not installed; skipping this example.")
}

# More examples can be found in demo `doRNG`
## Not run:
demo('doRNG')

## End(Not run)

## Some features of the %dorng% foreach operator
if (requireNamespace("doParallel", quietly = TRUE)) {
  ## Fork backend on Unix
  if (.Platform$OS.type == "unix") {
```

```
    doParallel::registerDoParallel(2)

    r1 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
    r2 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
    identical(r1, r2)
    attr(r1, "rng")

    set.seed(1234)
    s1 <- foreach(i=1:4) %dorng% { runif(1) }
    s2 <- foreach(i=1:4) %dorng% { runif(1) }
    identical(s1, s2)

    set.seed(1234)
    s1.2 <- foreach(i=1:4) %dorng% { runif(1) }
    s2.2 <- foreach(i=1:4) %dorng% { runif(1) }
    identical(s1, s1.2) && identical(s2, s2.2)
    identical(r1, s1)
  }

  ## PSOCK cluster (works everywhere)
  cl <- parallel::makeCluster(2)
  doParallel::registerDoParallel(cl)
  on.exit(parallel::stopCluster(cl), add = TRUE)

  s1 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
  s2 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
  identical(s1, s2)

  registerDoSEQ()
} else {
  message("Package 'doParallel' is not installed; skipping these examples.")
}

## Not run:
# MPI cluster (requires a working MPI + Rmpi setup)
library(doMPI)
cl <- startMPIcluster(2)
registerDoMPI(cl)

s1 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
s2 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }
identical(s1, s2)

closeCluster(cl)
registerDoSEQ()

## End(Not run)
```

---

doRNGversion *Back Compatibility Option for doRNG*

Description

Sets the behaviour of given version number.

Usage

```
doRNGversion(x)
```

Arguments

x	version number to switch to, or missing to get the currently active version number, or NULL to reset to the default behaviour, i.e. of the latest version.

Value

a character string If x is missing this function returns the version number from the current behaviour. If x is specified, the function returns the old value of the version number (invisible).

Behaviour changes in versions

1.4 The behaviour of doRNGseed, and therefore of ' L'Ecuyer-CMRG. Using set.seed before a non-seeded loop used not to be identical to seeding via .options.RNG. Another bug was that non-seeded loops would share most of their RNG seed!

1.7.4 Prior to this version, in the case where the RNG had not been called yet, the first seeded ' subsequent loops despite using the same seed (see https://github.com/renozao/doRNG/issues/12).

This has been fixed in version 1.7.4, where the RNG is called once (sample(NA)), whenever the .Random.seed is not found in global environment.

Examples

```
## Seeding when current RNG is L'Ecuyer-CMRG
RNGkind("L'Ecuyer")

doRNGversion("1.4")
# in version >= 1.4 seeding behaviour changed to fix a bug
set.seed(123)
res <- foreach(i=1:3) %dorng% runif(1)
res2 <- foreach(i=1:3) %dorng% runif(1)
stopifnot( !identical(attr(res, 'rng')[2:3], attr(res2, 'rng')[1:2]) )
res3 <- foreach(i=1:3, .options.RNG=123) %dorng% runif(1)
stopifnot( identical(res, res3) )

# buggy behaviour in version < 1.4
doRNGversion("1.3")
res <- foreach(i=1:3) %dorng% runif(1)
res2 <- foreach(i=1:3) %dorng% runif(1)
stopifnot( identical(attr(res, 'rng')[2:3], attr(res2, 'rng')[1:2]) )
res3 <- foreach(i=1:3, .options.RNG=123) %dorng% runif(1)
```

```
stopifnot( !identical(res, res3) )

# restore default RNG
RNGkind("default")
# restore to current doRNG version
doRNGversion(NULL)
```

`registerDoRNG` *Registering doRNG for Persistent Reproducible Parallel Foreach Loops*

Description

`registerDoRNG` registers the doRNG foreach backend. Subsequent ‘ registered foreach backend, but are internally performed as [making them fully reproducible.

Usage

```
registerDoRNG(seed = NULL, once = TRUE)
```

Arguments

`seed`	a numerical seed to use (as a single or 6-length numerical value)
`once`	a logical to indicate if the RNG sequence should be seeded at the beginning of each loop or only at the first loop.

Details

Briefly, the RNG is set, before each iteration, with seeds for L’Ecuyer’s CMRG that overall generate a reproducible sequence of statistically independent random streams.

Note that (re-)registering a foreach backend other than doRNG, after a call to `registerDoRNG` disables doRNG – which then needs to be registered.

Value

The value returned by [foreach::setDoPar]

See Also

[

Examples

```
if (requireNamespace("doParallel", quietly = TRUE)) {
  cl <- parallel::makeCluster(2)
  doParallel::registerDoParallel(cl)

  tryCatch({
    # One can make reproducible loops using the %dorng% operator
    r1 <- foreach(i=1:4, .options.RNG=1234) %dorng% { runif(1) }

    # or convert %dopar% loops using registerDoRNG
    registerDoRNG(1234)
    r2 <- foreach(i=1:4) %dopar% { runif(1) }
    identical(r1, r2)

    # Registering another foreach backend disables doRNG
    doParallel::registerDoParallel(cl)
    set.seed(1234)
    s1 <- foreach(i=1:4) %dopar% { runif(1) }
    set.seed(1234)
    s2 <- foreach(i=1:4) %dopar% { runif(1) }
    identical(s1, s2)


    # doRNG is re-enabled by re-registering it
    registerDoRNG()
    set.seed(1234)
    r3 <- foreach(i=1:4) %dopar% { runif(1) }
    identical(r2, r3)

    # argument `once=FALSE` reseeds doRNG's seed at the beginning of each loop
    registerDoRNG(1234, once=FALSE)
    a1 <- foreach(i=1:4) %dopar% { runif(1) }
    a2 <- foreach(i=1:4) %dopar% { runif(1) }
    identical(a1, a2)

    # Once doRNG is registered the seed can also be passed as an option to %dopar%
    b1 <- foreach(i=1:4, .options.RNG=456) %dopar% { runif(1) }
    b2 <- foreach(i=1:4, .options.RNG=456) %dopar% { runif(1) }
    identical(b1, b2) && !identical(b1, r1)

  }, finally = {
    foreach::registerDoSEQ()
    parallel::stopCluster(cl)
  })
} else {
  message("Package 'doParallel' is not installed; skipping this example.")
}
```

Index