The SyncRNG package provides a random number generator implemented in C and linked to both R and Python. This way, you can generate the same random number sequence in both languages by using the same seed.

The package implements a Tausworthe LSFR RNG (more details at https://gertjanvandenburg.com/blog/syncrng/). This is a very fast pseudo-random number generator.
Usage

There are two ways to use this package in R. It can be used as a reference class, where a SyncRNG object is used to keep the state of the generator and numbers are generated using the object methods. It can also be used as a user-defined random number generator using the strategy outlined in Random.user. See the examples section below.

Author(s)

Gerrit J.J. van den Burg
Maintainer: Gerrit J.J. van den Burg <gertjanvandenburg@gmail.com>

References

URL: https://github.com/GjjvdBurg/SyncRNG

Examples

library(SyncRNG)

# As user defined RNG:

set.seed(0, 'user', 'user')
runif(2)
# [1] 3.666952e-04 6.257184e-05
set.seed(0, 'user', 'user')
rnorm(2)
# [1] 0.01006027 0.42889422

# As class:

s <- SyncRNG(seed=0)
s$rand()
# [1] 0.0003666952
s$rand()
# [1] 6.257184e-05

SyncRNG-class  A Reference Class for SyncRNG

Description

See syncrng-package for package documentation.

Fields

seed  The seed for the random number generator
state  The current state of the RNG, should not be modified by the user
SyncRNG-class

Methods

- initialize(...) Initialize the RNG using the C function `R_syncrng_seed`
- `rand()` Generate a single random float in the range [0, 1)
- `randbelow(n)` Generate a random integer below a given number
- `randi()` Generate a single random 32-bit integer
- `shuffle(x)` Randomly shuffle a provided array of values

Examples

```r
s <- SyncRNG(seed=123456)
for (i in 1:10)
  cat(s$randi(), 'n')
```

Index

SyncRNG (SyncRNG-class), 2
SyncRNG-class, 2
-syncrng-package, 1, 2