Package 'rngwell19937'

February 20, 2015

Title Random number generator WELL19937a with 53 or 32 bit output

Version 0.6-0

Date 2014-11-30

Author Petr Savicky <savicky@cs.cas.cz>

Maintainer Petr Savicky <savicky@cs.cas.cz>

Description Long period linear random number generator WELL19937a by F. Panneton, P. L'Ecuyer and M. Matsumoto. The initialization algorithm allows to seed the generator with a numeric vector of an arbitrary length and uses MRG32k5a by P. L'Ecuyer to achieve good quality of the initialization. The output function may be set to provide numbers from the interval (0,1) with 53 (the default) or 32 random bits. WELL19937a is of similar type as Mersenne Twister and has the same period. WELL19937a is slightly slower than Mersenne Twister, but has better equidistribution and ``bit-mixing'' properties and faster recovery from states with prevailing zeros than Mersenne Twister. All WELL generators with orders 512, 1024, 19937 and 44497 can be found in randtoolbox package.

License file LICENSE

NeedsCompilation yes

License_restricts_use yes

Repository CRAN

Date/Publication 2014-11-30 17:11:25

R topics documented:

rngwell19937-package *Random number generator of Mersenne-Twister type with resolution 53 bits*

Description

After loading the package, the default random number generator in R is replaced by the long period linear generator WELL19937a by F. Panneton, P. L'Ecuyer and M. Matsumoto with a transformation to produce numbers in (0,1) with 53 (the default) or 32 random bits. In particular, the functions `set.seed()`, `runif()` and the object `.Random.seed` refer to WELL19937a.

A call of `detach("package:rngwell19937", unload=TRUE)` restores the default random number generator in R.

References

F. Panneton, P. L'Ecuyer, and M. Matsumoto, "Improved Long-Period Generators Based on Linear Recurrences Modulo 2", ACM Transactions on Mathematical Software, 32, 1 (2006), 1-16.

See Also

`set.resolution`, `set.initialization`, `set.vector.seed`, `well19937.validate`

set.initialization *Set initialization algorithm*

Description

The function allows to specify the initialization algorithm used by the function `set.seed()`.

Usage

```
set.initialization(initialization)
```

Arguments

`initialization` Character, either "mrg32k5a" or "sfmt". The required initialization algorithm.

Details

Initialization "mrg32k5a" uses the generator MRG32k5a by P. L'Ecuyer, see the citation below. Initialization "sfmt" is the same as in SFMT generator by Mutsuo Saito (Hiroshima University) and Makoto Matsumoto (Hiroshima University).

Value

`NULL` invisibly.

References

P. L'Ecuyer, "Good Parameter Sets for Combined Multiple Recursive Random Number Generators", Shorter version in Operations Research, 47, 1 (1999), 159–164.

`set.resolution` *Set the resolution of the output numbers*

Description

The function allows to choose between resolution 53 and 32 bits. The resolution is the number of random bits in the numbers produced by `runif()`.

Usage

```
set.resolution(resolution)
```

Arguments

`resolution`	Numeric, 53 or 32. The required number of bits.

Details

The new setting of the resolution is valid for the next calls to `runif()`.

Generating a random number with the resolution 53 requires two iterations of the internal 32 bit generator and their outputs are combined to a single number. The slow down of 53 bit resolution compared to 32 bit resolution is about 20 percent, since an R call to `runif()` performs also other actions besides calling the internal generator and these actions are run only once.

Value

`NULL` invisibly.

`set.vector.seed` *Seeding the generator with a numeric vector*

Description

The function allows to seed the generator with a numeric vector of an arbitrary length.

Usage

```
set.vector.seed(seed)
```

Arguments

seed	A numeric or integer vector, whose components have integer values in the interval [0, 2^32-1].

Details

Function `set.vector.seed()` generalizes the initialization `"mrg32k5a"` for longer seeds.

The input numbers should have integer values in the interval `[0, 2^32 - 1]`. They may be represented by numeric data type (double precision). If some of the components of the vector `seed` is at least `2^31`, then the numeric type is necessary.

If `seed` has length 1 and `0 <= seed <= 2^31 - 1`, then `set.vector.seed(seed)` with `"mrg32k5a"` initialization is equivalent to `set.seed(seed)`. If `seed` has length 1 and `2^31 + 1 <= seed <= 2^32 - 1`, then `set.vector.seed(seed)` with `"mrg32k5a"` initialization is equivalent to `set.seed(seed - 2^32)`.

Every two different input vectors of length at most 3 produce different initial states of WELL19937a. The bit pattern of the seed is not important. In particular, the seeds 0, 1, or 2 are not worse than any other seed and produce unrelated sequences.

Value

`NULL` invisibly.

`well19937.validate` *Checking the installation of the generator*

Description

The generator is run with a fixed seed and its output is compared to stored precomputed values.

Usage

```
well19937.validate()
```

Details

Loading the package `rngwell19937` sets `RNGkind("user-supplied")`. If this setting is changed by the user before calling `well19937.validate()`, the function resets `RNGkind("user-supplied")`.

Value

The function prints the result of the test. The output value is `NULL` invisibly.

Examples

```
well19937.validate()
```

Index