Package 'PRNG'

August 17, 2024

Title A Pseudo-Random Number Generator

Version 0.0.2.1

Description Provides functions for generating pseudo-random numbers that follow a uniform distribution [0,1]. Randomness tests were conducted using the National Institute of Standards and Technology test suite<https://csrc.nist.gov/pubs/sp/800/22/r1/upd1/final>, along with additional tests. The sequence generated depends on the initial values and parameters. The package includes a linear congruence map as the decision map and three chaotic maps to generate the pseudo-random sequence, which follow a uniform distribution. Other distributions can be generated from the uniform distribution using the Inversion Principle Method and BOX-Muller transformation. Small perturbations in seed values result in entirely different sequences of numbers due to the sensitive nature of the maps being used. The chaotic nature of the maps helps achieve randomness in the generator. Additionally, the generator is capable of producing random bits.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0), nortest

Config/testthat/edition 3

NeedsCompilation no

Author Sajad Ahmad Mir [aut, cre] (<https://orcid.org/0009-0009-6422-1893>), Dr. Puneet Sharma [aut] (<https://orcid.org/0000-0001-7839-0051>)

Maintainer Sajad Ahmad Mir <mir.1@iitj.ac.in>

Repository CRAN

Date/Publication 2024-08-17 06:30:20 UTC

Contents

`baker_map` *Baker map*

Description

this is a chaotic map with the sensitive for the parameter value greater than 0.5

Usage

```
baker_map(x0, a)
```

Arguments

`x0`	seed value
`a`	parameter of the map range is greater than 0.5

Value

for 0<=x=<1/2 the map returns 2ax for 1/2<=x<=1 the map returns a(2x-1)mod1

Examples

```
baker_map(0.3,0.56)
```

`linear_con` *Linear congruence map*

Description

the map is a member of the family of the maps f(x)=(ax+b) mod(n)

Usage

```
linear_con(x0)
```

Arguments

`x0`	seed value

Value

the map gives an integer ax+b mod(n)

Examples

```
linear_con(5)
```

`logistic_map` *Logistic map*

Description

This is the most used chaotic map . The map is sensitive for the value of the parameter greater than 3.568

Usage

```
logistic_map(x0, a)
```

Arguments

`x0`	the seed value range from 0 to 1
`a`	the parameter ranging from 3.5 to 4

Value

the map returns the a*x(1-x) for input x

Examples

```
logistic_map(0.26,3.5)
```

rbits *Random Bit generator*

Description

this function generates random bits of desired length

Usage

```
rbits(n, Time = TRUE)
```

Arguments

n	number of bits required
Time	it is a boolean value of TRUE/FALSE if we want to generate time dependent random bits.i.e each time we call the function with same input different output will be generated.

Value

returns a vector of random bits of length n

Examples

```
rbits(2)
rbits(2)
rbits(2,Time=FALSE)
rbits(2,Time=FALSE)
 rbits(10)
```

rcauchy *Cauchy distribution*

Description

This function generates random numbers from standard cauchy distribution

Usage

```
rcauchy(n, Time = TRUE)
```

Arguments

n	How many numbers we want
Time	time dependent or not

Value

a vector of n numbers from cauchy distribution

Examples

```
rcauchy(10)
rcauchy(10,Time=TRUE)
rcauchy(10,Time=TRUE)
```

`rexp` *Exponentail distribution*

Description

This function generates random numbers from exponentail distribution

Usage

```
rexp(n, Time = TRUE)
```

Arguments

`n`	how many numbers we need
`Time`	time dependent or not

Value

a vector of n numbers from exponential distribution

Examples

```
rexp(10)
  rexp(10)
  rexp(10,FALSE)
  rexp(10,FALSE)
```

rnorm *Generating numbers form Normal distribution here we use Box Muler transform to obtain normal random variable*

Description

Generating numbers form Normal distribution here we use Box Muler transform to obtain normal random variable

Usage

```
rnorm(n)
```

Arguments

n	number required

Value

a list of pseudo random numbers from normal distribution

Examples

```
rnorm(10)
 rnorm(100)
```

runf *Uniformly Pseudo random number generator*

Description

this function generates random numbers which follow uniform distribution `[0,1]`

Usage

```
runf(
  N = 100,
  Time = TRUE,
  n0 = 5,
  x00 = 0.5362,
  x01 = 0.357,
  x02 = 0.235,
  a1 = 3.69,
  a2 = 0.7
)
```

Arguments

N	How many numbers are required
Time	if enabled TRUE the numbers are time dependent
n0	seed value of linear congruence map it can take value of any natural number
x00	seed value of saw-tooth map values from 0 to 1
x01	seed value of logistic map values from 0 to 1
x02	seed value of baker map
a1	parameter of logistic map the value takes from 3.5 to 4
a2	parameter of baker map the value it takes values greater than or equalt to 0.5

Value

gives a vector of pseudo random numbers generated of desired length

Examples

```
  runf(10)
runf(10,Time=TRUE)
runf(10,Time=TRUE)
runf(10,Time=TRUE)
runf(10,2)
runf(10,Time=TRUE,2)
runf(10,Time=TRUE,2)

runf(10,5,0.52)
 runf(15,2,0.352)

  runf(10,2,0.652,0.235)
   runf(10,Time=TRUE,2,0.652,0.235)
  runf(9,7,0.52,0.4235,0.389)
  runf(10,Time=TRUE,2,0.752,0.235,0.351,3.8)
```

---

saw_tooth *Saw tooth map*

Description

saw tooth map is a family of maps as f(x)=b*x mod1

Usage

```
saw_tooth(x0)
```

Arguments

x0	seed value ranging from 0 to 1

Value

(3*x) mod(1)

Examples

```
saw_tooth(0.6)
```

`time` *time function*

Description

This function is used to generate a time of the system to be used for generating time dependent random numbers precise upto micro-seconds

Usage

```
time()
```

Value

t fractional value of the time

Examples

```
time()
```

Index