# Package: sitmo (via r-universe)

October 28, 2026

**Title** Parallel Pseudo Random Number Generator (PPRNG) 'sitmo' Header Files

**Version** 2.0.2

**Description** Provided within are two high quality and fast PPRNGs that may be used in an 'OpenMP' parallel environment. In addition, there is a generator for one dimensional low-discrepancy sequence. The objective of this library to consolidate the distribution of the 'sitmo' (C++98 & C++11), 'threefry' and 'vandercorput' (C++11-only) engines on CRAN by enabling others to link to the header files inside of 'sitmo' instead of including a copy of each engine within their individual package. Lastly, the package contains example implementations using the 'sitmo' package and three accompanying vignette that provide additional information.

**Depends** R (>= 3.2.0)

**URL** https://github.com/coatless/sitmo, http://thecoatlessprofessor.com/projects/sitmo/, https://github.com/stdfin/random/

**BugReports** https://github.com/coatless/sitmo/issues

**License** MIT + file LICENSE

**LinkingTo** Rcpp

**Imports** Rcpp (>= 0.12.13)

**RoxygenNote** 7.1.1

**Suggests** knitr, rmarkdown, ggplot2

**VignetteBuilder** knitr

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** James Balamuta [aut, cre, cph] (<https://orcid.org/0000-0003-2826-8458>), Thijs van den Berg [aut, cph], Ralf Stubner [ctb]

**Maintainer** James Balamuta <balamut2@illinois.edu>

**Repository** https://cran.r-universe.dev

**Date/Publication** 2021-10-13 03:50:02 UTC

**RemoteUrl** https://github.com/cran/sitmo

**RemoteRef** HEAD

**RemoteSha** 0e5ae28f8a85fad5f8fb261e1e92167bf68efabf

# Contents

sitmo-package . . . . . 2
runif_r . . . . . 3
runif_sitmo . . . . . 3
sitmo_draws . . . . . 4
sitmo_engine_reset . . . . . 5
sitmo_engine_seed . . . . . 5
sitmo_parallel . . . . . 6
sitmo_two_seeds . . . . . 7

**Index** **8**

---

`sitmo-package` *sitmo: Parallel Pseudo Random Number Generator (PPRNG) 'sitmo' Header Files*

---

## Description

Provided within are two high quality and fast PPRNGs that may be used in an 'OpenMP' parallel environment. In addition, there is a generator for one dimensional low-discrepancy sequence. The objective of this library to consolidate the distribution of the 'sitmo' (C++98 & C++11), 'threefry' and 'vandercorput' (C++11-only) engines on CRAN by enabling others to link to the header files inside of 'sitmo' instead of including a copy of each engine within their individual package. Lastly, the package contains example implementations using the 'sitmo' package and three accompanying vignette that provide additional information.

## Author(s)

**Maintainer**: James Balamuta <balamut2@illinois.edu> (ORCID) [copyright holder]

Authors:

- Thijs van den Berg <thijs@sitmo.com> [copyright holder]

Other contributors:

- Ralf Stubner <ralf.stubner@daqana.com> [contributor]

## See Also

Useful links:

- https://github.com/coatless/sitmo
- http://thecoatlessprofessor.com/projects/sitmo/
- https://github.com/stdfin/random/
- Report bugs at https://github.com/coatless/sitmo/issues

---

`runif_r` *Random Uniform Number Generator using base R*

---

## Description

The function provides an alternative implementation of random uniform distribution sampling using R's rng scope.

## Usage

```
runif_r(n, min = 0, max = 1)
```

## Arguments

| | |
|---|---|
| `n` | An `unsigned integer` denoting the number of realizations to generate. |
| `min` | A `double` indicating the minimum $a$ value in the uniform's interval $[a, b]$ |
| `max` | A `double` indicating the maximum $b$ value in the uniform's interval $[a, b]$ |

## Examples

```
set.seed(134)
b = runif_r(10)
```

---

`runif_sitmo` *Random Uniform Number Generator with sitmo*

---

## Description

The function provides an implementation of sampling from a random uniform distribution

## Usage

```
runif_sitmo(n, min = 0, max = 1, seed = 1L)
```

### Arguments

| | |
|---|---|
| n | An unsigned integer denoting the number of realizations to generate. |
| min | A double indicating the minimum $a$ value in the uniform's interval $[a, b]$ |
| max | A double indicating the maximum $b$ value in the uniform's interval $[a, b]$ |
| seed | A special unsigned integer containing a single seed. |

### Value

A numeric vector containing the realizations.

### Examples

```
a = runif_sitmo(10)
```

---

## sitmo_draws *Example RNG Draws with sitmo*

---

### Description

Shows a basic setup and use case for sitmo.

### Usage

```
sitmo_draws(n)
```

### Arguments

| | |
|---|---|
| n | A unsigned int is a . |

### Value

A vec with random sequences.

### Examples

```
n = 10
a = sitmo_draws(n)
```

## sitmo_engine_reset *Example Seed Set and RNG Draws with sitmo*

### Description

Shows how to set a seed in sitmo.

### Usage

```
sitmo_engine_reset(n, seed)
```

### Arguments

| | |
|---|---|
| n | An unsigned int that dictates how many realizations occur. |
| seed | An unsigned int that controls the rng seed. |

### Value

A matrix with random sequences.

### Examples

```
n = 10
a = sitmo_engine_reset(n, 1337)

isTRUE(all.equal(a[,1],a[,2]))
```

## sitmo_engine_seed *Example Seed Set and RNG Draws with sitmo*

### Description

Shows how to set a seed in sitmo.

### Usage

```
sitmo_engine_seed(n, seed)
```

### Arguments

| | |
|---|---|
| n | An unsigned int that dictates how many realizations occur. |
| seed | An unsigned int that controls the rng seed. |

### Value

A vector with random sequences.

## Examples

```
n = 10
a = sitmo_engine_seed(n, 1337)
b = sitmo_engine_seed(n, 1337)
c = sitmo_engine_seed(n, 1338)

isTRUE(all.equal(a,b))
isTRUE(all.equal(a,c))
```

---

`sitmo_parallel` *Test Generation using sitmo and C++11*

---

## Description

The function provides an implementation of creating realizations from the default engine.

## Usage

```
sitmo_parallel(n, seeds)
```

## Arguments

| | |
|---|---|
| `n` | An `unsigned integer` denoting the number of realizations to generate. |
| `seeds` | A `vec` containing a list of seeds. Each seed is run on its own core. |

## Details

The following function's true power is only accessible on platforms that support OpenMP (e.g. Windows and Linux). However, it does provide a very good example as to how to make ones code applicable across multiple platforms.

With this being said, how we determine how many cores to split the generation to is governed by the number of seeds supplied. In the event that one is using OS X, only the first seed supplied is used.

## Value

A `vec` containing the realizations.

## Examples

```
a = sitmo_parallel(10, c(1))

b = sitmo_parallel(10, c(1,2))

c = sitmo_parallel(10, c(1,2))

# True on only OS X or systems without openmp
isTRUE(all.equal(a,b))
```

```
isTRUE(all.equal(b,c))
```

---

`sitmo_two_seeds` *Two RNG engines running side-by-side*

---

## Description

Shows how to create two separate RNGs and increase them together.

## Usage

```
sitmo_two_seeds(n, seeds)
```

## Arguments

| | |
|---|---|
| `n` | An `unsigned int` that dictates how many realizations occur. |
| `seeds` | A `vec` containing two integers greater than 0. |

## Value

A `matrix` with random sequences.

## Examples

```
n = 10
a = sitmo_two_seeds(n, c(1337, 1338))

b = sitmo_two_seeds(n, c(1337, 1337))

isTRUE(all.equal(a[,1], a[,2]))

isTRUE(all.equal(b[,1], b[,2]))

isTRUE(all.equal(a[,1], b[,1]))
```

# Index

runif_r, 3
runif_sitmo, 3

sitmo *(*sitmo-package*)*, 2
sitmo-package, 2
sitmo_draws, 4
sitmo_engine_reset, 5
sitmo_engine_seed, 5
sitmo_parallel, 6
sitmo_two_seeds, 7