

# Package ‘rstream’

November 17, 2009

**Version** 1.2.4

**Date** 2009-11-17

**Title** Streams of random numbers

**Author** Josef Leydold <leydold@statistik.wu-wien.ac.at>

**Maintainer** Josef Leydold <leydold@statistik.wu-wien.ac.at>

**Depends** R (>= 2.0.0), methods

**Description** Unified object oriented interface for multiple independent streams of random numbers from different sources.

**License** GPL-2

**URL** <http://statistik.wu-wien.ac.at/arvag/>

**Repository** CRAN

**Date/Publication** 2009-11-17 19:36:41

## R topics documented:

rstream-package . . . . .	2
rstream-class . . . . .	3
rstream.antithetic-methods . . . . .	5
rstream.clone-methods . . . . .	6
rstream.incprecision-methods . . . . .	7
rstream.lecuyer-class . . . . .	8
rstream.mrg32k3a-class . . . . .	11
rstream.name-methods . . . . .	14
rstream.packed-methods . . . . .	15
rstream.reset-methods . . . . .	16
rstream.resetsubstream-methods . . . . .	17
rstream.RNG . . . . .	18
rstream.runif-class . . . . .	19
rstream.sample-methods . . . . .	22

<b>Index</b>	<b>23</b>
--------------	-----------

---

rstream-package      *"rstream" – A package for multiple streams of uniform random numbers*

---

## Description

Unified interface to uniform random number generators

## Details

Package: rstream  
Type: Package  
Version: 1.2.4  
Date: 2009-11-17  
License: GPL 2 or later

This package provides a unified interface to uniform random number Objects of its classes act as a source of streams for random numbers that can be handled by a set of methods and which can be used as arguments of functions that require sources of randomness, e.g. discrete event models, Monte Carlo integration or other stochastic simulations.

The instances of objects are independent, i.e., sampling random numbers from one instance or resetting and changing its state does not effect any other instance. (However, different streams may not be *stochastically* independent if they are not carefully seeded!)

Additionally there is a mechanismus interface to save and restore such streams, e.g. for the next R session, for a reruning some stochastic calculations with identical input, or for sending the stream object to a slave node in parallel computing via MPI.

The interface to these classes is inspired by Pierre L'Ecuyers RngStreams package.

For an overview of classes and methods, see [rstream-class](#).

## Author(s)

Josef Leydold <leydold@statmath.wu-wien.ac.at>

## References

P. L'Ecuyer and J. Leydold (2005): rstream: Streams of Random Numbers for Stochastic Simulation, R News 5(2), 16–20.

## See Also

[rstream-class](#).

---

rstream-class      *Class "rstream" – Multiple streams of uniform random numbers*

---

## Description

The virtual class "rstream" provides a unified interface to uniform random number generators. Objects of its subclasses act as a source of streams for random numbers that can be handled by a set of methods and which can be used as arguments of functions that require sources of randomness, e.g. discrete event models, Monte Carlo integration or other stochastic simulations.

The instances of objects of this class and its subclasses are independent, i.e., sampling random numbers from one instance or resetting and changing its state does not effect any other instance. (However, different streams may not be *stochastically* independent if they are not carefully seeded!)

Additionally there is a mechanism interface to save and restore such streams, e.g. for the next R session, for a rerunning some stochastic calculations with identical input, or for sending the stream object to a slave node in parallel computing via MPI.

The interface to these classes is inspired by Pierre L'Ecuyers RngStreams package.

## Objects from the Class

A virtual Class: No objects may be created from it.

## Methods

The virtual class "rstream" prepares the following methods for handling random stream objects. Some methods that return parameters of the stream object have a variant that uses <- to change the respective parameters. See the man pages for the respective methods for details.

**Notice:** Some subclasses only implement a subset of these interfaces. The methods that do not work produce an error message.

Methods to use the stream (available for all subclasses):

**rstream.sample** signature(object = "rstream"): Get a random sample from the stream object.

**r** signature(object = "rstream"): Same as rstream.sample.

**rstream.reset** signature(object = "rstream"): Reset stream into initial state.

Some subclasses have implemented the concept of substreams. This is especially usefull if two or more streams should be synchronized:

**rstream.nextsubstream** signature(object = "rstream"): Set state of stream to next substream.

**rstream.resetsubstream** signature(object = "rstream"): Reset current substream into starting state.

Antithetic random streams return numbers which have smallest possible correlation (i.e. -1) to their respective counterparts:

**rstream.antithetic** signature(object = "rstream"): Whether or not the stream object returns antithetic random numbers.

**rstream.antithetic<-** signature(object = "rstream"): Change antithetic flag (TRUE or FALSE).

Most sources of pseudo random numbers generate random numbers of precision  $2^{-32} \approx 2 \times 10^{-10}$ . For some streams this can be increased to machine epsilon (i.e.  $\approx 10^{-16}$ ) by combining two random numbers of lower precision:

**rstream.increasedprecision** signature(object = "rstream"): Whether or not the stream object returns random numbers with increased precision.

**rstream.increasedprecision<-** signature(object = "rstream"): Change flag for increased precision (TRUE or FALSE).

Handling "rstream" objects:

**print** signature(x = "rstream"): Print state of the stream object.

**show** signature(x = "rstream"): Same as print.

**rstream.name** signature(object = "rstream"): The name of the stream object.

**rstream.name<-** signature(object = "rstream"): Change the name of the stream object.

**rstream.clone** signature(object = "rstream"): Make a copy (clone) of stream object.

When a "rstream" object should be used in another R session or saved for some kind of later reuse all information about the object must be packed. Notice no method other than unpacking can be applied to a packed object. It must be unpacked before.

**rstream.packed** signature(object = "rstream"): Whether or not the stream object is packed.

**rstream.packed<-** signature(object = "rstream"): Pack or unpack object: set packed to TRUE or FALSE.

### Warning

When "rstream" objects should be used in later R sessions they must be packed before the current R session is stopped and unpacked after the new R session has been started and the workspace image has been reloaded.

### Note

"rstream" objects cannot simply be copied by `<-`. The new variable does not hold a copy of an "rstream" object but just points to the old object which might not be the wanted result (similar to R environments). Use `rstream.clone` instead.

The actual interface is implemented in subclasses, one for each source (package/library) of random numbers. The slots of this class and of all its subclasses must not be accessed directly. Use the above methods instead.

One may miss a method for reseeding a random stream. However, there is no need for such a method as there is a method for resetting the stream to its initial state. I one needs a stream with a different stream, then a new rstream object should be created at all.

Packed objects must be unpacked before any other method can be applied.

### Author(s)

Josef Leydold (leydold@statistik.wu-wien.ac.at)

### References

L'Ecuyer, P., Simard, R., Chen, E. J., and Kelton, W. D. (2002) *An object-oriented random-number package with many long streams and substreams*. Operations Research 50(6), 1073-1075.

### See Also

[rstream.mrg32k3a-class](#), [rstream.runif-class](#), [rstream.antithetic-methods](#), [rstream.clone-methods](#), [rstream.incprecision-methods](#), [rstream.name-methods](#), [rstream.packed-methods](#), [rstream.reset-methods](#), [rstream.sample-methods](#), [rstream.nextsubstream-methods](#), [rstream.RNG](#).

---

rstream.antithetic-methods

*Methods for Function rstream.antithetic in Package 'rstream'*

---

### Description

Get and change the antithetic flag for an "rstream" object in package **rstream**.

If the antithetic flag is on (TRUE) the "rstream" object returns antithetic random numbers.

Antithetic random streams return numbers which have smallest possible correlation (i.e.  $-1$ ) to their respective counterparts. Thus instead of a number  $u$  the number  $1 - u$  is returned.

This is useful for variance reduction techniques in Monte Carlo computations.

### Usage

```
## S4 method for signature 'rstream':
rstream.antithetic(stream)
rstream.antithetic(stream) <- value
```

### Arguments

stream            an "rstream" object.

value            a boolean (TRUE or FALSE) to change the status of the flag.

## Methods

Methods available for the following "rstream" subclasses: `rstream.mrg32k3a-class`, `rstream.runif-class`.

## See Also

`rstream-class`.

## Examples

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## set antithetic flag of rstream object
rstream.antithetic(s) <- TRUE

## get antithetic flag of rstream object
rstream.antithetic(s)
```

---

rstream.clone-methods

*Methods for Function rstream.clone in Package 'rstream'*

---

## Description

Make a copy (clone) of an "rstream" object in package **rstream**.

"rstream" objects cannot simply be copied by `<-`. The new variable does not hold a copy of an "rstream" object but just points to the old object which might not be the wanted result (similar to R environments). To get a copy of the whole "rstream" object the clone method must be used.

## Usage

```
clone <- rstream.clone(stream)
```

## Arguments

`stream`            an "rstream" object.

## Methods

Methods available for all "rstream" subclasses: `rstream.mrg32k3a-class`, `rstream.runif-class`.

## Note

The label (name) of the new copy has a dot "." appended to distinguish the original object from its copy.

**See Also**

[rstream-class](#).

**Examples**

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## make a copy (clone)
clone <- rstream.clone(s)
```

---

rstream.incprecision-methods

*Methods for Function rstream.incprecision in Package 'rstream'*

---

**Description**

Get and change the flag for increased precision of an "rstream" object in package **rstream**.

If the increased precision flag is on (TRUE) the "rstream" object returns random numbers with precision close to machine epsilon.

Most sources of pseudo random numbers generate random numbers of precision  $2^{-32} \approx 2 \times 10^{-10}$ . When the flag is on the precision is increased to machine epsilon (i.e.  $\approx 10^{-16}$ ) by combining two random numbers of default precision.

**Usage**

```
## S4 method for signature 'rstream':
rstream.incprecision(stream)
rstream.incprecision(stream) <- value
```

**Arguments**

stream	an "rstream" object.
value	a boolean (TRUE or FALSE) to change the status of the flag.

**Methods**

Methods available for the following "rstream" subclasses: [rstream.mrg32k3a-class](#).

**See Also**

[rstream-class](#).

**Examples**

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## set increased precision flag of rstream object
rstream.incprecision(s) <- TRUE

## get increased precision flag of rstream object
rstream.incprecision(s)
```

---

```
rstream.lecuyer-class
```

*Class "rstream.lecuyer" – Multiple streams with MRG32k3a backbone generator from Pierre L'Ecuyers RngStreams package*

---

**Description**

This class is obsolete and should be replaced by class "rstream.mrg32k3a"!

This is the old class that implements the "rstream" interface for Pierre L'Ecuyer's RngStreams package with the MRG32K3a multiple recursive generator as its backbone generator. This package is well suited for multiple independent streams of uniform random numbers. In particular it provides antithetic variates and substreams. A very important feature is that different streams are stochastically independent (in opposition to many other random number generators where the user has to choose seeds carefully). For that reason there only exists a package seed for all streams and individual seeds should be avoided (and requires expertise).

**Objects from the Class**

Objects can be created by calls of the form `new("rstream.lecuyer", name, seed, force.seed, antithetic, incprecision)`.

**name:** An arbitrary string to name the stream object. If omitted a string that consists of `lecuyer` and some number (which is increased every time when a new *rstream* object is created).

**seed:** An array of six numbers. The seed for the RngStreams package. If omitted a random seed is used. It should only be set at the first creation of an instance of an *rstream.lecuyer* object.

**force.seed:** A boolean. If the RngStreams package should be reseeded (which is not recommended) it must be `TRUE`. Default is `FALSE`.

**antithetic:** A boolean. Whether or not antithetic random numbers should be produced. Default is `FALSE`.

**incprecision:** A boolean. Whether or not random numbers with increased precision should be produced. Default is `FALSE`.

**Extends**

Class "rstream", directly.

## Methods

The class "rstream.lecuyer" provides the following methods for handling "rstream.lecuyer" objects. Some methods that return parameters of the stream object have a variant that uses <- to change the respective parameters. See the man pages for the respective methods for details.

Methods to use the stream:

**rstream.sample** signature(object = "rstream.lecuyer"): Get a random sample from the stream object.

**r** signature(object = "rstream.lecuyer"): Same as rstream.sample.

**rstream.reset** signature(object = "rstream.lecuyer"): Reset stream into initial state.

**rstream.nextsubstream** signature(object = "rstream.lecuyer"): Set state of stream to next substream.

**rstream.resetsubstream** signature(object = "rstream.lecuyer"): Reset current substream into starting state.

Antithetic random streams and increased precision:

**rstream.antithetic** signature(object = "rstream.lecuyer"): Whether or not the stream object returns antithetic random numbers.

**rstream.antithetic<-** signature(object = "rstream.lecuyer"): Change antithetic flag (TRUE or FALSE).

**rstream.increprecision** signature(object = "rstream.lecuyer"): Whether or not the stream object returns random numbers with increased precision.

**rstream.increprecision<-** signature(object = "rstream.lecuyer"): Change flag for increased precision (TRUE or FALSE).

Handling "rstream.lecuyer" objects:

**print** signature(x = "rstream.lecuyer"): Print state of the stream object.

**rstream.name** signature(object = "rstream.lecuyer"): The name of the stream object.

**rstream.name<-** signature(object = "rstream.lecuyer"): Change the name of the stream object.

**rstream.clone** signature(object = "rstream.lecuyer"): Make a copy (clone) of stream object.

**initialize** signature(.Object = "rstream.lecuyer"): Initialize rstream object. (For Internal usage only).

When a "rstream.lecuyer" object should be used in another R session or saved for some kind of later reuse all information about the object must be packed. Notice no method other than unpacking can be applied to a packed object. It must be unpacked before.

**rstream.packed** signature(object = "rstream.lecuyer"): Whether or not the stream object is packed.

**rstream.packed<-** signature(object = "rstream.lecuyer"): Pack or unpack object: set packed to TRUE or FALSE.

**Warning**

The underlying RngStreams library uses a global variable to store the package seed. This variable is also stored inside R. Whenever a new instance of a "rstream.lecuyer" object is created the value of global variable is set to the value of the R object. Thus there is no problem when such "rstream.lecuyer" objects are packed for using in later R sessions. However, if such packed objects are not stored in the workspace image, then the R variable gets lost and there is a (extremely small) chance that newly created objects are not stochastically independent from restored objects.

**Note**

The slots of this class must not be accessed directly. Use the above methods instead.

"rstream" objects cannot simply be copied by `<-`. The new variable does not hold a copy of an "rstream" object but just points to the old object which might not be the wanted result (similar to R environments). Use `rstream.clone` instead.

One may miss a method for reseeding a random stream. However, there is no need for such a method as there is a method for resetting the stream to its initial state. I one needs a stream with a different stream, then a new rstream object should be created at all.

Packed objects must be unpacked before any other method can be applied.

**Author(s)**

Josef Leydold (leydold@statistik.wu-wien.ac.at)

underlying generator: Pierre L'Ecyuer and Richard Simard

**References**

L'Ecyuer, P., Simard, R., Chen, E. J., and Kelton, W. D. (2002) *An object-oriented random-number package with many long streams and substreams*. Operations Research 50(6), 1073-1075.

**See Also**

[rstream-class](#), [rstream.mrg32k3a-class](#), [rstream.antithetic-methods](#), [rstream.clone-methods](#), [rstream.incprecision-methods](#), [rstream.name-methods](#), [rstream.packed-methods](#), [rstream.reset-methods](#), [rstream.sample-methods](#), [rstream.nextsubstream-methods](#), [rstream.RNG](#).

**Examples**

```
## create a new rstream.lecuyer object
s <- new("rstream.lecuyer")

## show state of this object
print(s)

## show and change name of stream object
rstream.name(s)
rstream.name(s) <- "mystream"

## get a random number
```

```

x <- rstream.sample(s)

## get a random sample of size 100
x <- rstream.sample(s,100)

## reset random stream
rstream.reset(s)

## show and set antithetic flag
rstream.antithetic(s)
rstream.antithetic(s) <- TRUE

## jump to next substream
rstream.nextsubstream(s)

## make a clone of the rstream object
sc <- rstream.clone(s)

## pack and unpack the rstream object
rstream.packed(s) <- TRUE
rstream.packed(s) <- FALSE

```

---

```
rstream.mrg32k3a-class
```

*Class "rstream.mrg32k3a" – Multiple streams with MRG32k3a backbone generator from Pierre L'Ecuyers RngStreams package*

---

## Description

This class implements the "rstream" interface for Pierre L'Ecuyer's RngStreams package with the MRG32K3a multiple recursive generator as its backbone generator. This package is well suited for multiple independent streams of uniform random numbers. In particular it provides antithetic variates and substreams. A very important feature is that different streams are stochastically independent (in opposition to many other random number generators where the user has to choose seeds carefully). For that reason there only exists a package seed for all streams and individual seeds should be avoided (and requires expertise).

## Objects from the Class

Objects can be created by calls of the form `new("rstream.mrg32k3a", name, seed, force.seed, antithetic, inprecision)`.

**name:** An arbitrary string to name the stream object. If omitted a string that consists of `mrg32k3a` and some number (which is increased every time when a new *rstream* object is created).

**seed:** An array of six numbers. The seed for the RngStreams package. If omitted a random seed is used. It should only be set at the first creation of an instance of an *rstream.mrg32k3a* object.

**force.seed:** A boolean. If the RngStreams package should be reseeded (which is not recommended) it must be `TRUE`. Default is `FALSE`.

**antithetic:** A boolean. Whether or not antithetic random numbers should be produced. Default is `FALSE`.

**incprecision:** A boolean. Whether or not random numbers with increased precision should be produced. Default is `FALSE`.

## Extends

Class "rstream", directly.

## Methods

The class "rstream.mrg32k3a" provides the following methods for handling "rstream.mrg32k3a" objects. Some methods that return parameters of the stream object have a variant that uses `<-` to change the respective parameters. See the man pages for the respective methods for details.

Methods to use the stream:

**rstream.sample** signature(object = "rstream.mrg32k3a"): Get a random sample from the stream object.

**r** signature(object = "rstream.mrg32k3a"): Same as `rstream.sample`.

**rstream.reset** signature(object = "rstream.mrg32k3a"): Reset stream into initial state.

**rstream.nextsubstream** signature(object = "rstream.mrg32k3a"): Set state of stream to next substream.

**rstream.resetsubstream** signature(object = "rstream.mrg32k3a"): Reset current substream into starting state.

Antithetic random streams and increased precision:

**rstream.antithetic** signature(object = "rstream.mrg32k3a"): Whether or not the stream object returns antithetic random numbers.

**rstream.antithetic<-** signature(object = "rstream.mrg32k3a"): Change antithetic flag (`TRUE` or `FALSE`).

**rstream.incprecision** signature(object = "rstream.mrg32k3a"): Whether or not the stream object returns random numbers with increased precision.

**rstream.incprecision<-** signature(object = "rstream.mrg32k3a"): Change flag for increased precision (`TRUE` or `FALSE`).

Handling "rstream.mrg32k3a" objects:

**print** signature(x = "rstream.mrg32k3a"): Print state of the stream object.

**rstream.name** signature(object = "rstream.mrg32k3a"): The name of the stream object.

**rstream.name<-** signature(object = "rstream.mrg32k3a"): Change the name of the stream object.

**rstream.clone** signature(object = "rstream.mrg32k3a"): Make a copy (clone) of stream object.

**initialize** signature(.Object = "rstream.mrg32k3a"): Initialize rstream object. (For Internal usage only).

When a "rstream.mrg32k3a" object should be used in another R session or saved for some kind of later reuse all information about the object must be packed. Notice no method other than unpacking can be applied to a packed object. It must be unpacked before.

**rstream.packed** signature(object = "rstream.mrg32k3a"): Whether or not the stream object is packed.

**rstream.packed<-** signature(object = "rstream.mrg32k3a"): Pack or unpack object: set packed to TRUE or FALSE.

### Warning

The underlying RngStreams library uses a global variable to store the package seed. This variable is also stored inside R. Whenever a new instance of a "rstream.mrg32k3a" object is created the value of global variable is set to the value of the R object. Thus there is no problem when such "rstream.mrg32k3a" objects are packed for using in later R sessions. However, if such packed objects are not stored in the workspace image, then the R variable gets lost and there is a (extremely small) chance that newly created objects are not stochastically independent from restored objects.

### Note

The slots of this class must not be accessed directly. Use the above methods instead.

"rstream" objects cannot simply be copied by <-. The new variable does not hold a copy of an "rstream" object but just points to the old object which might not be the wanted result (similar to R environments). Use `rstream.clone` instead.

One may miss a method for reseeding a random stream. However, there is no need for such a method as there is a method for resetting the stream to its initial state. I one needs a stream with a different stream, then a new rstream object should be created at all.

Packed objects must be unpacked before any other method can be applied.

### Author(s)

Josef Leydold (leydold@statistik.wu-wien.ac.at)

underlying generator: Pierre L'Ecyuer and Richard Simard

### References

L'Ecyuer, P., Simard, R., Chen, E. J., and Kelton, W. D. (2002) *An object-oriented random-number package with many long streams and substreams*. Operations Research 50(6), 1073-1075.

### See Also

[rstream-class](#), [rstream.antithetic-methods](#), [rstream.clone-methods](#), [rstream.inprecision-methods](#), [rstream.name-methods](#), [rstream.packed-methods](#), [rstream.reset-methods](#), [rstream.sample-methods](#), [rstream.nextsubstream-methods](#), [rstream.RNG](#).

## Examples

```
## create a new rstream.mrg32k3a object
s <- new("rstream.mrg32k3a")

## show state of this object
print(s)

## show and change name of stream object
rstream.name(s)
rstream.name(s) <- "mystream"

## get a random number
x <- rstream.sample(s)

## get a random sample of size 100
x <- rstream.sample(s,100)

## reset random stream
rstream.reset(s)

## show and set antithetic flag
rstream.antithetic(s)
rstream.antithetic(s) <- TRUE

## jump to next substream
rstream.nextsubstream(s)

## make a clone of the rstream object
sc <- rstream.clone(s)

## pack and unpack the rstream object
rstream.packed(s) <- TRUE
rstream.packed(s) <- FALSE
```

---

rstream.name-methods

*Methods for Function rstream.name in Package 'rstream'*

---

## Description

Get and change the name for an "rstream" object in package **rstream**.

The name is a character string that gives a user the possibility to label an "rstream" object. Any name can be used.

## Usage

```
## S4 method for signature 'rstream':
rstream.name(stream)
rstream.name(stream) <- value
```

## Arguments

`stream` an "rstream" object.  
`value` a character string that holds the label (name).

## Methods

Methods available for all "rstream" subclasses: `rstream.mrg32k3a-class`, `rstream.runif-class`.

## See Also

`rstream-class`.

## Examples

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## set name of rstream object
rstream.name(s) <- "mystream"

## get name of rstream object
rstream.name(s)
```

---

rstream.packed-methods

*Methods for Function rstream.packed in Package 'rstream'*

---

## Description

Get status (packed/unpacked) of an "rstream" object in package **rstream**.

Pack and unpack an "rstream" object in package **rstream**.

When a "rstream" object should be used in another R session or saved for some kind of later reuse all information about the object must be packed. Notice no method other than unpacking can be applied to a packed object. It must be unpacked before.

## Usage

```
## S4 method for signature 'rstream':
rstream.packed(stream)
rstream.packed(stream) <- value
```

**Arguments**

stream            an "rstream" object.  
 value            a boolean (TRUE or FALSE) to change the status of the object.  
                  TRUE: pack the object.  
                  FALSE: unpack the object.

**Methods**

Methods available for all "rstream" subclasses: `rstream.mrg32k3a-class`, `rstream.runif-class`.

**See Also**

`rstream-class`.

**Examples**

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## pack rstream object
rstream.packed(s) <- TRUE

## status of object
rstream.packed(s)

## pack rstream object
rstream.packed(s) <- FALSE
```

---

`rstream.reset-methods`

*Methods for Function `rstream.reset` in Package 'rstream'*

---

**Description**

Reset "rstream" object in package **rstream** into its initial state.

By resetting the stream object the same stream of random numbers can be generated.

**Usage**

```
## S4 method for signature 'rstream':
rstream.reset(stream)
```

**Arguments**

stream            an "rstream" object.

**Methods**

Methods available for all "rstream" subclasses: `rstream.mrg32k3a-class`, `rstream.runif-class`.

**See Also**

`rstream-class`.

**Examples**

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## generate a sample
x <- rstream.sample(s,10)

## reset the stream object
rstream.reset(s)

## the new sample y is identical to x
y <- rstream.sample(s,10)
if (identical(x,y)) print("x and y are identical samples")
```

---

rstream.resetsubstream-methods

*Methods for Function rstream.resetsubstream and  
rstream.nextsubstream in Package 'rstream'*

---

**Description**

Some subclasses have implemented the concept of substreams. This is especially useful if two or more streams should be synchronized.

This interface allows to jump to the beginning of the next substream and to reset the stream object to the beginning of the current substream.

**Usage**

```
## S4 method for signature 'rstream':
rstream.resetsubstream(stream)
## S4 method for signature 'rstream':
rstream.nextsubstream(stream)
```

**Arguments**

`stream` an "rstream" object.

## Methods

Methods available for the following "rstream" subclasses: [rstream.mrg32k3a-class](#).

## See Also

[rstream-class](#).

## Examples

```
## create a new rstream object (of subclass rstream.mrg32k3a)
s <- new("rstream.mrg32k3a")

## jump to beginning of next substream
rstream.nextsubstream(s)

## generate a sample
x <- rstream.sample(s,10)

## reset substream
rstream.resetsubstream(s)

## the new sample y is identical to x
y <- rstream.sample(s,10)
if (identical(x,y)) print("x and y are identical samples")
```

---

rstream.RNG

*Get and set "rstream" object from/to R global generator*

---

## Description

The function `rstream.RNG(stream)` is used to set a given "rstream" object `stream` as current global R uniform random number generators.

Without an argument (or `NULL`) it returns an "rstream" object that contains the current global generator. It is a copy (clone) of the global generator and thus it can be handled independently from the global generator.

## Usage

```
rstream.RNG(stream = NULL)
```

## Arguments

`stream`            `NULL` or an "rstream" object

## Value

`rstream.RNG` returns an "rstream" object.

**Author(s)**

Josef Leydold (leydold@statistik.wu-wien.ac.at)

**See Also**

[rstream-class](#).

**Examples**

```
## create a new rstream.runif object
s <- new("rstream.mrg32k3a")

## use this stream as global R uniform RNG
rstream.RNG(s)

## get a (independent) copy of the stream
## that contains the global R uniform RNG
gs <- rstream.RNG()

## change the state of the global generator
gs <- rstream.RNG()
rstream.nextsubstream(gs)
rstream.RNG(gs)
```

---

```
rstream.runif-class
```

*Class "rstream.runif" – Interface to R internal uniform random number generators*

---

**Description**

This class implements the "rstream" interface for the R internal uniform RNGs. This class allows to access and handle these generators in exactly the same way as external generators. In particular, one can create copies of generators. There is no need to deal with `RNGkind` and `set.seed` when different RNGs should be used.

**Objects from the Class**

Objects can be created by calls of the form `new("rstream.runif", name, kind, seed, antithetic)`.

**name:** An arbitrary string to name the stream object. If omitted a string that consists of `runif` and some number (which is increased every time when a new *rstream* object is created).

**kind:** A character string. The new "rstream" object uses the RNG of type `kind`. The same strings as for `RNGkind` can be used. Additionally the string `"current"` is available to use the type of generator to which R is currently set (this is the default if no `kind` is given). User-supplied generators cannot be used.

**seed:** The seed for the generator as used by the `set.seed` call. If omitted a random seed is used.

**antithetic:** A boolean. Whether or not antithetic random numbers should be produced. Default is `FALSE`.

## Extends

Class "rstream", directly.

## Methods

The class "rstream.runif" provides the following methods for handling "rstream.runif" objects. Some methods that return parameters of the stream object have a variant that uses `<-` to change the respective parameters. See the man pages for the respective methods for details.

Methods to use the stream:

**rstream.sample** signature(object = "rstream.runif"): Get a random sample from the stream object.

**r** signature(object = "rstream.runif"): Same as `rstream.sample`.

**rstream.reset** signature(object = "rstream.runif"): Reset stream into initial state.

Antithetic random streams:

**rstream.antithetic** signature(object = "rstream.runif"): Whether or not the stream object returns antithetic random numbers.

**rstream.antithetic<-** signature(object = "rstream.runif"): Change antithetic flag (TRUE or FALSE).

Handling "rstream.runif" objects:

**print** signature(x = "rstream.runif"): Print state of the stream object.

**rstream.name** signature(object = "rstream.runif"): The name of the stream object.

**rstream.name<-** signature(object = "rstream.runif"): Change the name of the stream object.

**rstream.clone** signature(object = "rstream.runif"): Make a copy (clone) of stream object.

**initialize** signature(.Object = "rstream.runif"): Initialize rstream object. (For Internal usage only).

When a "rstream.runif" object should be used in another R session or saved for some kind of later reuse all information about the object must be packed. Notice no method other than unpacking can be applied to a packed object. It must be unpacked before.

**rstream.packed** signature(object = "rstream.runif"): Whether or not the stream object is packed.

**rstream.packed<-** signature(object = "rstream.runif"): Pack or unpack object: set packed to TRUE or FALSE.

**Note**

The slots of this class must not be accessed directly. Use the above methods instead.

"rstream" objects cannot simply be copied by `<-`. The new variable does not hold a copy of an "rstream" object but just points to the old object which might not be the wanted result (similar to R environments). Use `rstream.clone` instead.

One may miss a method for reseeding a random stream. However, there is no need for such a method as there is a method for resetting the stream to its initial state. If one needs a stream with a different stream, then a new rstream object should be created at all.

Packed objects must be unpacked before any other method can be applied.

**Author(s)**

Josef Leydold (leydold@statistik.wu-wien.ac.at)

**See Also**

[rstream-class](#), [rstream.antithetic-methods](#), [rstream.clone-methods](#), [rstream.name-methods](#), [rstream.packed-methods](#), [rstream.reset-methods](#), [rstream.sample-methods](#), [rstream.RNG](#).

**Examples**

```
## create a new rstream.runif object
s <- new("rstream.runif")

## show state of this object
print(s)

## show and change name of stream object
rstream.name(s)
rstream.name(s) <- "mystream"

## get a random number
x <- rstream.sample(s)

## get a random sample of size 100
x <- rstream.sample(s,100)

## reset random stream
rstream.reset(s)

## show and set antithetic flag
rstream.antithetic(s)
rstream.antithetic(s) <- TRUE

## make a clone of the rstream object
sc <- rstream.clone(s)

## pack and unpack the rstream object
rstream.packed(s) <- TRUE
```

```
rstream.packed(s) <- FALSE
```

---

```
rstream.sample-methods
```

*Methods for Function `rstream.sample` in Package ‘rstream’*

---

## Description

Get random sample from an "rstream" object in package **rstream**.

## Usage

```
## S4 method for signature 'rstream,numeric':  
rstream.sample(stream,n=1)  
## S4 method for signature 'rstream,numeric':  
r(stream,n=1)
```

## Arguments

<code>stream</code>	an "rstream" object.
<code>n</code>	sample size.

## Methods

Methods available for all "rstream" subclasses: [rstream.mrg32k3a-class](#), [rstream.runif-class](#).

## Note

`r` is equivalent to `rstream.sample`.

## See Also

[rstream-class](#).

## Examples

```
## create a new rstream object (of subclass rstream.mrg32k3a)  
s <- new("rstream.mrg32k3a")  
  
## get a random number  
x <- rstream.sample(s)  
  
## get a random sample of size 100  
x <- rstream.sample(s,100)  
  
## alternatively ...  
x <- r(s,100)
```

# Index

## \*Topic **classes**

- `rstream-class`, 2
- `rstream.lecuyer-class`, 8
- `rstream.mrg32k3a-class`, 11
- `rstream.runif-class`, 19

## \*Topic **datagen**

- `rstream-class`, 2
- `rstream-package`, 1
- `rstream.antithetic-methods`, 5
- `rstream.clone-methods`, 6
- `rstream.incprecision-methods`, 7
- `rstream.lecuyer-class`, 8
- `rstream.mrg32k3a-class`, 11
- `rstream.name-methods`, 14
- `rstream.packed-methods`, 15
- `rstream.reset-methods`, 16
- `rstream.resetsubstream-methods`, 17
- `rstream.RNG`, 18
- `rstream.runif-class`, 19
- `rstream.sample-methods`, 21

## \*Topic **distribution**

- `rstream-class`, 2
- `rstream-package`, 1
- `rstream.antithetic-methods`, 5
- `rstream.clone-methods`, 6
- `rstream.incprecision-methods`, 7
- `rstream.lecuyer-class`, 8
- `rstream.mrg32k3a-class`, 11
- `rstream.name-methods`, 14
- `rstream.packed-methods`, 15
- `rstream.reset-methods`, 16
- `rstream.resetsubstream-methods`, 17
- `rstream.RNG`, 18
- `rstream.runif-class`, 19
- `rstream.sample-methods`, 21

## \*Topic **methods**

- `rstream.antithetic-methods`, 5
- `rstream.clone-methods`, 6
- `rstream.incprecision-methods`, 7
- `rstream.name-methods`, 14
- `rstream.packed-methods`, 15
- `rstream.reset-methods`, 16
- `rstream.resetsubstream-methods`, 17
- `rstream.RNG`, 18
- `rstream.sample-methods`, 21

`initialize`, `rstream.lecuyer-method` (`rstream.lecuyer-class`), 8

`initialize`, `rstream.mrg32k3a-method` (`rstream.mrg32k3a-class`), 11

`initialize`, `rstream.runif-method` (`rstream.runif-class`), 19

`print`, `rstream-method` (`rstream-class`), 2

`print`, `rstream.lecuyer-method` (`rstream.lecuyer-class`), 8

`print`, `rstream.mrg32k3a-method` (`rstream.mrg32k3a-class`), 11

`print`, `rstream.runif-method` (`rstream.runif-class`), 19

`r` (`rstream.sample-methods`), 21

`r`, `rstream,numeric-method` (`rstream.sample-methods`), 21

`r`, `rstream-method` (`rstream.sample-methods`), 21

`r`, `rstream.lecuyer-method` (`rstream.sample-methods`), 21

- r, `rstream.mrg32k3a`-method  
(`rstream.sample-methods`),  
21
- r, `rstream.runif`-method  
(`rstream.sample-methods`),  
21
- r-methods  
(`rstream.sample-methods`),  
21
- RNGkind, 19
- `rstream` (`rstream-package`), 1
- `rstream`-class, 2, 5–7, 10, 13, 15–18, 21,  
22
- `rstream`-class, 2
- `rstream`-package, 1
- `rstream.antithetic`  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic`, `rstream`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic`, `rstream.lecuyer`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic`, `rstream.mrg32k3a`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic`, `rstream.runif`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic-methods`, 5, 10,  
13, 21
- `rstream.antithetic-methods`, 5
- `rstream.antithetic<=`  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic<=`, `rstream`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic<=`, `rstream.lecuyer`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic<=`, `rstream.mrg32k3a`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic<=`, `rstream.runif`-method  
(`rstream.antithetic-methods`),  
5
- `rstream.antithetic<=`-methods  
(`rstream.antithetic-methods`),  
5
- `rstream.clone`, 4, 10, 13, 20
- `rstream.clone`  
(`rstream.clone-methods`), 6
- `rstream.clone`, `rstream`-method  
(`rstream.clone-methods`), 6
- `rstream.clone`, `rstream.lecuyer`-method  
(`rstream.clone-methods`), 6
- `rstream.clone`, `rstream.mrg32k3a`-method  
(`rstream.clone-methods`), 6
- `rstream.clone`, `rstream.runif`-method  
(`rstream.clone-methods`), 6
- `rstream.clone-methods`, 5, 10, 13, 21
- `rstream.clone-methods`, 6
- `rstream.incprecision`  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision`, `rstream`-method  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision`, `rstream.lecuyer`-method  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision`, `rstream.mrg32k3a`-method  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision-methods`, 5,  
10, 13
- `rstream.incprecision-methods`, 7
- `rstream.incprecision<=`  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision<=`, `rstream`-method  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision<=`, `rstream.lecuyer`-method  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision<=`, `rstream.mrg32k3a`-method  
(`rstream.incprecision-methods`),  
7
- `rstream.incprecision<=`-methods  
(`rstream.incprecision-methods`),  
7
- `rstream.lecuyer`-class, 8
- `rstream.mrg32k3a`-class, 5–7, 10,

- 14–17, 22
- `rstream.mrg32k3a`-class, 11
- `rstream.name`  
(*rstream.name-methods*), 14
- `rstream.name`, `rstream`-method  
(*rstream.name-methods*), 14
- `rstream.name`, `rstream.lecuyer`-method  
(*rstream.name-methods*), 14
- `rstream.name`, `rstream.mrg32k3a`-method  
(*rstream.name-methods*), 14
- `rstream.name`, `rstream.runif`-method  
(*rstream.name-methods*), 14
- `rstream.name-methods`, 5, 10, 13, 21
- `rstream.name-methods`, 14
- `rstream.name`<-  
(*rstream.name-methods*), 14
- `rstream.name`<-, `rstream`-method  
(*rstream.name-methods*), 14
- `rstream.name`<-, `rstream.lecuyer`-method  
(*rstream.name-methods*), 14
- `rstream.name`<-, `rstream.mrg32k3a`-method  
(*rstream.name-methods*), 14
- `rstream.name`<-, `rstream.runif`-method  
(*rstream.name-methods*), 14
- `rstream.name`<-methods  
(*rstream.name-methods*), 14
- `rstream.nextsubstream`  
(*rstream.resetsubstream-methods*), 17
- `rstream.nextsubstream`, `rstream`-method  
(*rstream.resetsubstream-methods*), 17
- `rstream.nextsubstream`, `rstream.lecuyer`-method  
(*rstream.resetsubstream-methods*), 17
- `rstream.nextsubstream`, `rstream.mrg32k3a`-method  
(*rstream.resetsubstream-methods*), 17
- `rstream.nextsubstream-methods`, 5, 10, 13
- `rstream.nextsubstream-methods`  
(*rstream.resetsubstream-methods*), 17
- `rstream.packed`  
(*rstream.packed-methods*), 15
- `rstream.packed`, `rstream`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`, `rstream.lecuyer`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`, `rstream.mrg32k3a`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`, `rstream.runif`-method  
(*rstream.packed-methods*), 15
- `rstream.packed-methods`, 5, 10, 13, 21
- `rstream.packed-methods`, 15
- `rstream.packed`<-  
(*rstream.packed-methods*), 15
- `rstream.packed`<-, `rstream`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`<-, `rstream.lecuyer`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`<-, `rstream.mrg32k3a`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`<-, `rstream.runif`-method  
(*rstream.packed-methods*), 15
- `rstream.packed`<-methods  
(*rstream.packed-methods*), 15
- `rstream.reset`  
(*rstream.reset-methods*), 16
- `rstream.reset`, `rstream`-method  
(*rstream.reset-methods*), 16
- `rstream.reset`, `rstream.lecuyer`-method  
(*rstream.reset-methods*), 16
- `rstream.reset`, `rstream.mrg32k3a`-method  
(*rstream.reset-methods*), 16
- `rstream.reset`, `rstream.runif`-method  
(*rstream.reset-methods*), 16
- `rstream.reset-methods`, 5, 10, 13, 21
- `rstream.reset-methods`, 16
- `rstream.resetsubstream`  
(*rstream.resetsubstream-methods*), 17
- `rstream.resetsubstream`, `rstream`-method  
(*rstream.resetsubstream-methods*), 17

`rstream.resetsubstream`, `rstream.lecuyer`-method  
(*rstream.resetsubstream-methods*),  
[17](#)

`rstream.resetsubstream`, `rstream.mrg32k3a`-method  
(*rstream.resetsubstream-methods*),  
[17](#)

`rstream.resetsubstream-methods`,  
[17](#)

`rstream.RNG`, [5](#), [10](#), [13](#), [18](#), [21](#)

`rstream.runif-class`, [5](#), [6](#), [14–16](#), [22](#)

`rstream.runif-class`, [19](#)

`rstream.sample`  
(*rstream.sample-methods*),  
[21](#)

`rstream.sample`, `rstream`, numeric-method  
(*rstream.sample-methods*),  
[21](#)

`rstream.sample`, `rstream`-method  
(*rstream.sample-methods*),  
[21](#)

`rstream.sample`, `rstream.lecuyer`-method  
(*rstream.sample-methods*),  
[21](#)

`rstream.sample`, `rstream.mrg32k3a`-method  
(*rstream.sample-methods*),  
[21](#)

`rstream.sample`, `rstream.runif`-method  
(*rstream.sample-methods*),  
[21](#)

`rstream.sample-methods`, [5](#), [10](#), [13](#), [21](#)

`rstream.sample-methods`, [21](#)

`set.seed`, [19](#)

`show`, `rstream`-method  
(*rstream-class*), [2](#)