Package 'setRNG'

April 4, 2006

Title Set (normal) random number generator and seed

Description Set reproducible random number generator in R (and S). See ?setRNG.Intro for more details.

Depends R (>= 2.0.0)

Version 2006.4-1

LazyLoad yes

License Free. See the LICENCE file for details.

Author Paul Gilbert <pgilbert@bank-banque-canada.ca>

Maintainer Paul Gilbert <pgilbert@bank-banque-canada.ca>

URL http://www.bank-banque-canada.ca/pgilbert

R topics documented:

`00.setRNG.Intro` *setRNG*

Description

Programs to set random number generator (and seed) in R and S.

Details

See `setRNG-package` (in the help system use package?setRNG or ?"setRNG-package") for an overview.

`getRNG` *get the RND and seed from an object*

Description

Get the random number generator and seed used to generate an object.

Usage

```
getRNG(e=NULL)
## Default S3 method:
getRNG(e=NULL)
```

Arguments

`e`	an object generated by simulation (which stored the RNG information).

Details

Extract the RNG information used to generate the object. If `e` is `NULL` then `getRNG` returns the RNG setting, as returned by `setRNG()`. Otherwise, the default method assumes the object is a list and the RNG information is in the element `rng` or `noise$rng`.

Value

The random seed and other random number generation information used to generate the object.

See Also

`setRNG`, `.Random.seed`

Examples

```
## Not run:
if (require("dse1")) {
  data("eg1.DSE.data.diff", package="dse1")
  model <- estVARXls(eg1.DSE.data.diff)
  sim <- simulate(model)
  getRNG(sim)
  }
## End(Not run)
```

`random.number.test` *Test the random number generator*

Description

Test the random number generator.

Usage

```
random.number.test()
```

Arguments

None

Details

This function checks that the RNG is working properly and has not been changed. If the RNG does not return values as in previous versions of R then the function executes `stop()`. Since changes to the RNG will cause comparisons of simulation results to fail, this is a useful check before investigating more complicated problems that may be a result of other "improvements" in your simulation or estimation programs.

Value

logical

Side Effects

Executes `stop()` if the tests fail.

See Also

`set.seed` `RNGkind` `runif` `rnorm` `setRNG`

Examples

```
random.number.test()
```

`setRNG-package` *setRNG*

Description

Programs to set random number generator (and seed) in R and S.

Usage

library("setRNG")

Introduction

This library provides tools to simplify recording and resetting the random number generator, to help make monte carlo experiments easily reproducible. It uses the R/S tools for setting the seed, but also records and sets the mechanism for converting uniform numbers to normally distributed numbers. (It could be extended to other transformations, but I have not done that.)

The setRNG function would typically be called by simulation programs (see example) to set the RNG information if given, and record the RNG information in all cases. This information can be returned with the result of the simulation. That way the simulation can always be reproduced if necessary.

The library also implements an approach to random number generation which allows the same random experiments to be replicated in S and R. The functions in the S/ directory allow the R results using Wichmann-Hill and Box-Muller to be replicated in S. These were done with the aid of an example from B. D. Ripley. (The files in the S/ directory of the package are for use with S not R.) These functions are intended primarily as a way to confirm that simulations and estimations with simulated data work in the same way in both S and R, not as an improved RNG. (It has only been tested in Splus 3.3) Default and other RNGs can still be used and are preferred for both speed and theoretical reasons.

Examples

```
setRNG(kind="Wichmann-Hill", seed=c(979,1479,1542), normal.kind="Box-Muller")
rnorm(10)

sim <-function(rng=NULL)
 {if(!require("setRNG")) stop("This function requires the setRNG package.")
  if(is.null(rng)) rng <- setRNG() # returns setting so don't skip if NULL
  else        {old.rng <- setRNG(rng);  on.exit(setRNG(old.rng))  }
  x <- list(numbers=rnorm(10))
  x$rng <- rng
  x
 }

z <- sim()
sim()$numbers
sim(rng=getRNG(z))$numbers
z$numbers
```

`setRNG` *Set the Random Number Generator*

Description

Set the RNG or return information about the setting of the RNG.

Usage

```
setRNG(kind=NULL, seed=NULL, normal.kind=NULL)
```

Arguments

None required

`kind`	a character string.
`seed`	a vector of numbers (depending on kind).
`normal.kind`	a character string.

Details

Sets the uniform and normal random number generators and the seed. The old setting is returned in a format which can be used in another call to setRNG. (This would reset to the original value.) If no arguments are given the current setting is returned. In R see RNGkind for more details.

Value

The old setting.

Side Effects

Sets global variables controlling the uniform and normal random number generators and the global seed.

See Also

`RNGkind`, `set.seed`, `runif`, `rnorm`, `random.number.test`

Examples

```
setRNG(kind="Wichmann-Hill", seed=c(979,1479,1542), normal.kind="Box-Muller")
rnorm(10)
```

Index