Package 'qrandom'

October 13, 2022

Type Package

Title True Random Numbers using the ANU Quantum Random Numbers Server

Version 1.2.6

Date 2022-08-30

Maintainer Siegfried Köstlmeier <siegfried.koestlmeier@gmail.com>

Description

The ANU Quantum Random Number Generator provided by the Australian National University generates true random numbers in real-time by measuring the quantum fluctuations of the vacuum. This package offers an interface using their API.
The electromagnetic field of the vacuum exhibits random fluctuations in phase and amplitude at all frequencies.
By carefully measuring these fluctuations, one is able to generate ultra-high bandwidth random numbers.
The quantum Random Number Generator is based on the papers by Symul et al., (2011) <doi:10.1063/1.3597793>
and Haw, et al. (2015) <doi:10.1103/PhysRevApplied.3.054004>.
The package offers functions to retrieve a sequence of random integers or hexadecimals and true random samples from a normal or uniform distribution.

License GPL-2 | file LICENSE

Imports curl, jsonlite, methods, Rmpfr, utils

Suggests testthat

Encoding UTF-8

LazyData true

URL https://qrng.anu.edu.au/

NeedsCompilation no

Author Siegfried Köstlmeier [aut, cre]
(<https://orcid.org/0000-0002-7221-6981>),
Boris Steipe [ctb] (<https://orcid.org/0000-0002-1134-6758>)

Repository CRAN

Date/Publication 2022-08-30 08:00:02 UTC

R topics documented:

`qrandom` *Raw sequence of true random numbers by measuring the quantum fluctuations of the vacuum*

Description

`qrandom` implements an interface to the ANU Quantum Random Number Generator provided by the Australian National University. An ultra-high bandwith of true random numbers is generated in real-time by measuring the quantum fluctuations of the vacuum.

Usage

```
qrandom(n = 1, type = "uint8", blocksize = 1)
```

Arguments

`n`	The amount of random numbers to return. Must be between 1 and 100,000.
`type`	The data type must be `uint8`, `uint16` or `hex16`.
`blocksize`	Only needed for data type `hex16`. Sets the length of each block and has to be between 1 and 1,024.

Details

`qrandom` is based on the official QRNG@ANU API. The data type

- `uint8` returns uniformly distributed integers from the interval $[0; 255]$.
- `uint16` returns uniformly distributed integers from the interval $[0; 65,535]$.
- `hex16` returns uniformly distributed hexadecimal characters from the interval $[00; ff]$ for `blocksize = 1`.

Each request can return a minimum of 1 and a maximum of 100,000 true random numbers. The parameter `blocksize` is only relevant for data type `hex16` and sets the length of each block. `blocksize` must be between 1 and 1,024. A request with data type `"hex16"` returns hexadecimal characters with class `class character` and type `character`. For numeric calculation with these characters `randomHexCharacters`, they have to be coerced e.g. with `base::as.hexmode(randomHexCharacters)` for smaller hexadecimal numbers, or e.g. with `Rmpfr::mpfr(randomHexCharacters, base = 16)` for arbitrary precision floating point numbers.

The true random numbers are generated in real-time by measuring the quantum fluctuations of the vacuum. The official QRNG@ANU API supports only a maximum of 1,024 random numbers per request, thus requests for more numbers have to split up into smaller requests of 1,024 numbers. In fact, each request may take a couple of seconds to be served. The greatest possible request `qrandom(n = 100000, type = "hex16", blocksize = 1024)` takes about 13 minutes (via DSL 16,000 internet connection) and its size is about 201.4 MB.

We try our best to provide unique true random numbers. All API requests provided by this package are using SSL. As long as nobody is able to break the encryption protocol, the random numbers you obtain should be unique and secure.

Further information can be obtained from the ANU Quantum Random Number Generator FAQ and the list of references to scientific papers.

Value

`qrandom` returns an object of `class` `"integer"` for data type `uint8` or `uint16` and an object of class `"character"` for data type `hex16`. For numeric calculations, the hexadecimal characters have to be coerced to an appropriate class prior to calculations (e.g. with `base::as.hexmode` or `Rmpfr::mpfr`).

The returning value of `"qrandom"` is a `"vector"` containing true random numbers.

References

Secure Quantum Communication group, Centre for Quantum Computing and Communication Technology, Department of Quantum Sciences, Research School of Physics and Engineering, The Australian National University, Canberra, ACT 0200, Australia. *Welcome to the ANU Quantum Random Numbers Server.* https://qrng.anu.edu.au/

Examples

```
## request for 10 true random numbers within the interval [0; 255]
randomNumbers <- qrandom(n = 10)

## request for 10 true random numbers within the interval [0; 65,535]
randomNumbers <- qrandom(n = 10, type = "uint16")

## request for 10 true random hexadecimal characters within the interval [00; ff]
randomNumbers <- qrandom(n = 10, type = "hex16")

## requests with data type 'hex16' are characters
class(randomNumbers)
typeof(randomNumbers)

## request for 10 true random hexadecimal numbers within the interval [0000; ffff]
randomNumbers <- qrandom(n = 10, type = "hex16", blocksize = 2)

## coerce hexadecimal random characters to hexadecimal numbers prior to numeric calculations
## calculate mean of randomNumbers
library(Rmpfr)
mean(as.hexmode(randomNumbers))
```

```
mean(mpfr(randomNumbers, base = 16))
```

`qrandommaxint` *Uniformly distributed true random signed integers in the full range*

Description

`qrandommaxint` generates uniformly distributed true random signed integers from the API of the vacuum quantum-fluctuation server at the Australian National University. The numbers are uniformly distributed over the full range of signed integers, where both values are included.

Usage

```
qrandommaxint(n = 1)
```

Arguments

`n`	The number of random signed integers to return. Must be between 1 and 100,000. Default is 1.

Details

`qrandommaxint` is based on the function `qrandom` to generate input of n 32-bit numbers in hexadecimal representation. These numbers are converted to bit-representations and then interpreted as signed integers. This approach does not suffer from the sampling bias inherent in multiplying an integer range with a number from $U(0, 1)$.

The true random numbers are generated in real-time by measuring the quantum fluctuations of the vacuum. The official QRNG@ANU API currently supports only a maximum of 1,024 random numbers per request, thus requests for more numbers are split up into smaller requests of maximum 1,024 numbers. In fact, each request may take a couple of seconds to be served. The greatest possible request `qrandommaxint(n = 100000)` takes about 7 minutes (via DSL 16,000 internet connection) and its size is about 781.3 KB. The sequence of uniformly distributed hexadeximal octets is transformed into signed integers, which usually corresponds to $U(-2,147,483,647, 2,147,483,647)$. These numbers can be used to initialize R's RNG.

Although it is of little practical importance, note that both input patterns `0x00000000` and `0x80000000` are internally interpreted as zero, whereas `strtoi("0x80000000")` (and all larger, i.e. negative numbers) return `NA`.

For further information on the underlying function to retrieve quantum random numbers, see qrandom.

Further information can be obtained from the ANU Quantum Random Number Generator FAQ and the list of references to scientific papers.

Value

`qrandommaxint` returns an integer vector of length n `class` `"integer"` with type `"integer"` with n true random numbers, uniformly distributed over the full range of signed integers, where both values are included.

References

Secure Quantum Communication group, Centre for Quantum Computing and Communication Technology, Department of Quantum Sciences, Research School of Physics and Engineering, The Australian National University, Canberra, ACT 0200, Australia. *Welcome to the ANU Quantum Random Numbers Server.* https://qrng.anu.edu.au/

See Also

qrandom

Examples

```
## request for 10 true random integers
randomIntegers <- qrandommaxint(n = 10)

## randomly set the RNG seed
set.seed(qrandommaxint())
```

`qrandomnorm` *Normal distributed sample of true random numbers*

Description

`qrandomnorm` generates normal distributed true random numbers in real-time by measuring the quantum fluctuations of the vacuum. Per default, the data follows a standard normal distribution $N(0, 1)$, with mean zero and standard deviation one.

Usage

```
qrandomnorm(n = 1, mean = 0, sd = 1, method = "inverse")
```

Arguments

`n`	The amount of random numbers to return. Must be between 1 and 100,000.
`mean`	The mean-value for the normal distribution.
`sd`	The standard deviation for the normal distribution.
`method`	The method used for transforming internally used standard uniformly distributed data into normal distributed data. Currently provided methods are 'inverse' for inverse transform sampling using stats::qnorm(), 'polar' for the polar-method by George Marsaglia and 'boxmuller' for the Box-Muller transformation by George Box and Mervin Muller.

Details

qrandomnorm is based on the function qrandom to generate a sequence of a minimum of 1 and a maximum of 100,000 true random numbers. The true random numbers are generated in real-time by measuring the quantum fluctuations of the vacuum. The official QRNG@ANU API currently supports only a maximum of 1,024 random numbers per request, thus requests for more numbers have to split up into smaller requests of maximum 1,024 numbers. In fact, each request may take a couple of seconds to be served. The greatest possible requests `qrandomnorm(n = 100000, method = "polar")` or `qrandomnorm(n = 100000, method = "boxmuller")` take about 8 minutes (via DSL 16,000 internet connection) and their size is about 781.3 KB. Per default, the sequence of numbers is transformed into a standard normal distribution $N(0, 1)$, with $mean = 0$ and standard deviation $sd = 1$.

Internally, uniformly distributed true random numbers within the interval [0; 1] from the function qrandomunif are used to obtain normal distributed data. Within these uniformly data, the smallest possible number greater than zero is 2.220446e-16 and the largest possible number less than one is 0.999999999999997779554.

We provide three methods to transform our standard uniformly data $U(0, 1)$ into a normal distribution $N(mean, sd)$:

- inverseThe sample of standard uniformly data is interpreted as a probability and transformed into a normal distribution applying the stats::qnorm() function.
- polarUsing the polar-method by George Marsaglia for generating normal distributed data.
- boxmullerUsing the Box-Muller transformation by George Box and Mervin Muller for generating normal distributed data.

Be aware that only the default method 'inverse' is able to return -Inf and +Inf z-values for the normal distribution. The following table summarizes the non-infinite minimum and maximum z-values for a standard normal distribution for each method provided and compares them with the non-infinite extreme values from stats::qnorm():

method	stats:qnorm()	inverse	polar	boxmuller
minimum z-value*	-8.209536	-8.12589	-8.36707	-8.490424
maximum z-value*	8.209536	8.12589	8.36707	8.490424
z-values +-Inf	Yes	Yes	No	No

*non-infinite values.

We try our best to provide unique true random numbers. All API requests provided by this package are using SSL. As long as nobody is able to break the encryption protocol, the random numbers you obtain should be unique and secure.

Further information can be obtained from the ANU Quantum Random Number Generator FAQ and the list of references to scientific papers.

Value

qrandomnorm returns an object of `class` `"numeric"` with type `"double"`.

The returning value of `"qrandomnorm"` is a `"vector"` containing true random numbers which are normally distributed $N(mean, sd)$.

References

Benjamin, April 25th, 2017, answer on teo93, "generate N(0,1) using uniform(0,1) in R", Stack Overflow, January 4th 2009, https://stackoverflow.com/a/43619239/8512077.

Box, G. E. P. and Muller, Mervin E. (1958). A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, **29**, No. 2, p. 610-–611, doi:10.1214/aoms/1177706645.

Marsaglia, G. and Bray, T. A. (1964): A Convenient Method for Generating Normal Variables. *SIAM Review*, **6**, No. 3, p. 260-–264, doi:10.1137/1006063.

Secure Quantum Communication group, Centre for Quantum Computing and Communication Technology, Department of Quantum Sciences, Research School of Physics and Engineering, The Australian National University, Canberra, ACT 0200, Australia. *Welcome to the ANU Quantum Random Numbers Server.* https://qrng.anu.edu.au/

Wikipedia contributors. (2018, November 2). Inverse transform sampling. In Wikipedia, The Free Encyclopedia. Retrieved 13:03, January 4, 2019, from https://en.wikipedia.org/w/index.php?title=Inverse_transform_samplin

Wikipedia contributors. (2018, December 15). Box–Muller transform. In Wikipedia, The Free Encyclopedia. Retrieved 12:55, January 4, 2019, from https://en.wikipedia.org/w/index.php?title=Box-3Muller_transform&oldid=873905617.

Wikipedia contributors. (2018, November 29). Marsaglia polar method. In Wikipedia, The Free Encyclopedia. Retrieved 12:57, January 4, 2019, from https://en.wikipedia.org/w/index.php?title=Marsaglia_polar_method&old

See Also

qrandomunif

Examples

```
## request for 10 true standard normal distributed random numbers
randomNumbers <- qrandomnorm(n = 10)

## request for 10 true random numbers with mean 5 and standard deviation of 3
randomNumbers <- qrandomnorm(n = 10, mean = 5, sd = 3)

## request for 10 true random numbers with mean 8, standard deviation of 2 and using
## the polar-method by George Marsaglia to transform internally used uniformly data
## into a normal distribution
randomNumbers <- qrandomnorm(n = 10, mean = 8, sd = 2, method = "polar")

## calculate mean of randomNumbers
mean(randomNumbers)
```

---

qRandomSeq *Demo data of true random numbers*

Description

qRandomSeq is a sample of 1,000 true random numbers, which are generated by measuring the quantum fluctuations of the vacuum. The sequence of numbers is provided by the ANU Quantum Random Number Generator of the Australian National University.

Usage

```
data(qRandomSeq)
```

Details

qRandomSeq is a list containing 1,000 observations of true random numbers for each of the data-types provided by the official QRNG@ANU API. The data type

- uint8 returns uniformly distributed integers from the interval $[0; 255]$.
- uint16 returns uniformly distributed integers from the interval $[0; 65,535]$.
- hex16 returns uniformly distributed hexadecimal characters from the interval $[00; ff]$.

Examples

```
## load qRandomSeq
data(qRandomSeq)

## calculate the mean of the 1,000 random numbers
## with data-type uint8
mean(qRandomSeq$uint8)
```

---

qrandomunif *Uniformly distributed sample of true random numbers*

Description

qrandomunif generates uniformly distributed true random numbers in real-time by measuring the quantum fluctuations of the vacuum. Per default, the data follows a standard uniform distribution $U(0, 1)$ with minimum value $a = 0$ and maximum value $b = 1$, where both values are included.

Usage

```
qrandomunif(n = 1, a = 0, b = 1)
```

Arguments

n	The amount of random numbers to return. Must be between 1 and 100,000.
a	Minimum value restriction for the uniform distribution (inclusive a).
b	Maximum value restriction for the uniform distribution (inclusive b).

Details

`qrandomunif` is based on the function `qrandom` to generate a sequence of a minimum of 1 and a maximum of 100,000 true random numbers. The true random numbers are generated in real-time by measuring the quantum fluctuations of the vacuum. The official QRNG@ANU API currently supports only a maximum of 1,024 random numbers per request, thus requests for more numbers have to split up into smaller requests of maximum 1,024 numbers. In fact, each request may take a couple of seconds to be served. The greatest possible request `qrandomunif(n = 100000)` takes about 7 minutes (via DSL 16,000 internet connection) and its size is about 781.3 KB. The sequence of numbers is transformed into a uniformly distribution $U(a, b)$, where $a = 0$ and $b = 1$ are its minimum and maximum values (both included and $a < b$ is required).

Internally, hexadecimal numbers with block-size 7 are obtained via qrandom() and each of their first (random) character is deleted. These numbers are uniformly distributed within the range [0x0000000000000; 0xfffffffffffff] which is [0; 4,503,599,627,370,495] in decimal integers. As we already have uniformly distributed numbers, we just divide each number by the maximum possible value of 0xfffffffffffff to normalize our random numbers. After conversion into decimal numbers we obtain uniformly distributed numbers within the interval [0; 1]. Our procedure garanties, that the smallest possible number greater than zero is 2.220446e-16 and the largest possible number less than one is 0.999999999999997779554. For further information on why these boundaries are chosen, see qrandomnorm.

We try our best to provide unique true random numbers. All API requests provided by this package are using SSL. As long as nobody is able to break the encryption protocol, the random numbers you obtain should be unique and secure.

Further information can be obtained from the ANU Quantum Random Number Generator FAQ and the list of references to scientific papers.

Value

`qrandomunif` returns an object of `class` `"numeric"` with type `"double"`.

The returning value of `"qrandomunif"` is a `"vector"` containing true random numbers which are uniformly distributed $U(a, b)$.

References

Secure Quantum Communication group, Centre for Quantum Computing and Communication Technology, Department of Quantum Sciences, Research School of Physics and Engineering, The Australian National University, Canberra, ACT 0200, Australia. *Welcome to the ANU Quantum Random Numbers Server.* https://qrng.anu.edu.au/

See Also

qrandomnorm

Examples

```
## request for 10 true standard uniformed random numbers
randomNumbers <- qrandomunif(n = 10)
```

```
## request for 10 true random numbers within the interval [-10; 10]
randomNumbers <- qrandomunif(n = 10, a = -10, b = 10)

## calculate mean of randomNumbers
mean(randomNumbers)
```

---

qUUID *True random UUIDs conforming to RFC 4122*

Description

qUUID generates true random UUIDs conforming to RFC 4122 from random numbers retrieved through the API of the vacuum quantum-fluctuation server at the Australian National University. The UUIDs are taken from a space of `2^122 == 5.3e+36` possibilities.

Usage

```
qUUID(n = 1)
```

Arguments

n	The number of UUIDs to return. Must be between 1 and 100,000. Default is 1.

Details

qUUID returns true random Universally Unique IDentifiers (UUID) that conform to RFC 4122. The construction is based on the function qrandom to generate an input of n 128-bit numbers in hexadecimal representation. These numbers are converted to bit-representations, the 6-bit RFC 4122-required pattern for random UIDs is stamped onto these, and the result is converted back to hexadecimals giving `2^122 == 5.3e+36` possibilities. The resulting UUIDs are patterned like `"e3da74ea-b88d-fea2-0b69-56a16caeda0a"`; bits 61 to 64 are `(00 01 00 00)`, and bits 71 to 72 are `(00 01)` (counting from 1).

The true random source is generated in real-time by measuring the quantum fluctuations of the vacuum. The official QRNG@ANU API currently supports only a maximum of 1,024 random numbers per request, thus requests for more numbers are split up into smaller requests of maximum 1,024 numbers. In fact, each request may take a couple of seconds to be served.

For further information on the underlying function to retrieve quantum random numbers, see qrandom.

Further information can be obtained from the ANU Quantum Random Number Generator FAQ and the list of references to scientific papers.

For the UUID specification see RFC 4122.

Value

qUUID returns a character vector of length n, `class "character"`, type `"character"` containing n true random UUIDs conforming to RFC 4122.

References

Secure Quantum Communication group, Centre for Quantum Computing and Communication Technology, Department of Quantum Sciences, Research School of Physics and Engineering, The Australian National University, Canberra, ACT 0200, Australia. *Welcome to the ANU Quantum Random Numbers Server.* https://qrng.anu.edu.au/

Network Working Group (2005), *A Universally Unique IDentifier (UUID) URN Namespace (RFC 4122).* https://tools.ietf.org/html/rfc4122

See Also

qrandom

Examples

```
## generate five true random UUIDs
myUUIDs <- qUUID(5)
```

Index