Package 'qs'

May 17, 2019

Type Package

Title Quick Serialization of R Objects

Version 0.16.1

Date 2019-5-15

Maintainer Travers Ching <traversc@gmail.com>

Description Provides functions for quickly writing and reading any R object to and from disk.

License AGPL-3 | file LICENSE

LazyData true

Imports Rcpp, RApiSerialize

LinkingTo Rcpp, RApiSerialize

Depends R (>= 3.5.0)

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Copyright This package includes code from the 'zstd' library owned by Facebook, Inc. and created by Yann Collet; the 'lz4' library created by Yann Collet; and code derived from the 'Blosc' library created by Francesc Alted.

URL https://github.com/traversc/qs

BugReports https://github.com/traversc/qs/issues

NeedsCompilation yes

Author Travers Ching [aut, cre, cph],
Yann Collet [ctb, cph] (Yann Collet is the author of the bundled zstd and lz4 code),
Facebook, Inc. [cph] (Facebook is the copyright holder of the bundled zstd code),
Reichardt Tino [ctb, cph] (Contributor/copyright holder of zstd bundled code),
Skibinski Przemyslaw [ctb, cph] (Contributor/copyright holder of zstd bundled code),

Mori Yuta [ctb, cph] (Contributor/copyright holder of zstd bundled code),
Romain Francois [ctb, cph] (Derived example/tutorials for Alt-Rep structures),
Francesc Alted [ctb, cph] (Shuffling routines derived from Blosc library)

Repository CRAN

Date/Publication 2019-05-17 18:30:02 UTC

R topics documented:

`blosc_shuffle_raw` *Shuffle a raw vector*

Description

A function for shuffling a raw vector using BLOSC shuffle routines

Usage

```
blosc_shuffle_raw(x, bytesofsize)
```

Arguments

`x`	The raw vector
`bytesofsize`	Either 4 or 8

Value

The shuffled vector

Examples

```
x <- serialize(1L:1000L, NULL)
xshuf <- blosc_shuffle_raw(x, 4)
xunshuf <- blosc_unshuffle_raw(xshuf, 4)
```

`blosc_unshuffle_raw` *Un-shuffle a raw vector*

Description

A function for un-shuffling a raw vector using BLOSC un-shuffle routines

Usage

```
blosc_unshuffle_raw(x, bytesofsize)
```

Arguments

`x`	The raw vector
`bytesofsize`	Either 4 or 8

Value

The unshuffled vector

Examples

```
x <- serialize(1L:1000L, NULL)
xshuf <- blosc_shuffle_raw(x, 4)
xunshuf <- blosc_unshuffle_raw(xshuf, 4)
```

convertToAlt *Convert character vector to alt-rep*

Description

A function for generating a alt-rep object from a character vector, for users to experiment with the alt-rep system.

Usage

```
convertToAlt(x)
```

Arguments

x	The character vector

Value

The character vector in alt-rep form

Examples

```
xalt <- convertToAlt(randomStrings(N=10, string_size=20))
xalt2 <- convertToAlt(c("a", "b", "c"))
```

is_big_endian *System Endianness*

Description

Tests system endianness. Intel and AMD based systems are little endian, and so this function will likely return 'FALSE'. The 'qs' package is not capable of transferring data between systems of different endianness. This should not matter for the large majority of use cases.

Usage

```
is_big_endian()
```

Value

'TRUE' if big endian, 'FALSE' if little endian.

Examples

```
is_big_endian() # returns FALSE on Intel/AMD systems
```

lz4_compress_bound *lz4 compress bound*

Description

Exports the compress bound function from the lz4 library. Returns the maximum compressed size of an object of length ‘size‘.

Usage

```
lz4_compress_bound(size)
```

Arguments

size	An integer size

Value

maximum compressed size

Examples

```
lz4_compress_bound(100000)
#' lz4_compress_bound(1e9)
```

lz4_compress_raw *lz4 compression*

Description

Compression of raw vector. Exports the main lz4 compression function.

Usage

```
lz4_compress_raw(x, compress_level)
```

Arguments

x	A Raw Vector
compress_level	The compression level (> 1).

Value

The compressed data

Examples

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- lz4_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(lz4_decompress_raw(xcompressed))
```

`lz4_decompress_raw` *lz4 decompression*

Description

Decompresses of raw vector

Usage

```
lz4_decompress_raw(x)
```

Arguments

x	A Raw Vector

Value

The uncompressed data

Examples

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- lz4_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(lz4_decompress_raw(xcompressed))
```

---

qdump *qdump*

Description

Exports the uncompressed binary serialization to a list of Raw Vectors. For testing purposes and exploratory purposes mainly.

Usage

```
qdump(file)
```

Arguments

file	the file name/path.

Value

The uncompressed serialization

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
x2 <- qdump(myfile)
```

---

qinspect *qinspect*

Description

Performs a quick inspection of a serialized object/file, determines whether the file was properly compressed. E.g., if your process was interrupted for some reason, and you suspect qsave was interrupted, you can run this function to test the integrity of the serialized object.

Usage

```
qinspect(file)
```

Arguments

file	the file name/path

Value

A boolean. TRUE if the object was properly compressed. FALSE if there is an issue.

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
qinspect(myfile) # returns true
```

qread *qread*

Description

Reads a object in a file serialized to disk

Usage

```
qread(file, use_alt_rep=TRUE, inspect=FALSE, nthreads=1)
```

Arguments

file	the file name/path
use_alt_rep	Use alt rep when reading in string data. Default: TRUE
inspect	Whether to call qinspect before de-serializing data. Set to true if you suspect your data may be corrupted. Default: FALSE
nthreads	Number of threads to use. Default 1.

Value

The de-serialized object

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
x2 <- qread(myfile)
identical(x, x2) # returns true

# qs support multithreading
qsave(x, myfile, nthreads=2)
x2 <- qread(myfile, nthreads=2)
identical(x, x2) # returns true

# Other examples
z <- 1:1e7
myfile <- tempfile()
qsave(z, myfile)
z2 <- qread(myfile)
identical(z, z2) # returns true

w <- as.list(rnorm(1e6))
myfile <- tempfile()
qsave(w, myfile)
```

```
w2 <- qread(myfile)
identical(w, w2) # returns true
```

---

qsave *qsave*

Description

Saves (serializes) an object to disk.

Usage

```
qsave(x, file,
preset = "balanced", algorithm = "lz4", compress_level = 1L,
shuffle_control = 15L, nthreads = 1)
```

Arguments

x	the object to serialize.
file	the file name/path.
preset	One of "fast", "balanced" (default), "high" or "custom". See details.
algorithm	Compression algorithm used. Either lz4 (default or zstd).
compress_level	The compression level used (Default 1). For lz4, this number must be > 1 (higher is less compressed). For zstd, a number between -50 to 22 (higher is more compressed).
shuffle_control	An integer setting the use of byte shuffle compression. A value between 0 and 15 (Default 3). See details.
nthreads	Number of threads to use. Default 1.

Details

This function serializes and compresses R objects using block compresion with the option of byte shuffling. There are lots of possible parameters. This function exposes three parameters related to compression level and byte shuffling.

'compress_level' - Higher values tend to have a better compression ratio, while lower values/negative values tend to be quicker. Due to the format of qs, there is very little benefit to compression levels > 5 or so.

'shuffle_control' - This sets which numerical R object types are subject to byte shuffling. Generally speaking, the more ordered/sequential an object is (e.g., '1:1e7'), the larger the potential benefit of byte shuffling. It is not uncommon to have several orders magnitude benefit to compression ratio or compression speed. The more random an object is (e.g., 'rnorm(1e7)'), the less potential benefit there is, even negative benefit is possible. Integer vectors almost always benefit from byte shuffling whereas the results for numeric vectors are mixed. To control block shuffling, add +1

to the parameter for logical vectors, +2 for integer vectors, +4 for numeric vectors and/or +8 for complex vectors.

The ‘preset‘ parameter has several different combination of parameter sets that are performant over a large variety of data. The ‘algorithm‘ parameter, ‘compression_level‘ and ‘shuffle_control‘ parameters are ignored unless ‘preset‘ is "custom". "fast" preset: algorithm lz4, compress_level 100, shuffle_control 0. "balanced" preset: algorithm lz4, compress_level 1, shuffle_control 15. "high" preset: algorithm zstd, compress_level 5, shuffle_control 15.

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
x2 <- qread(myfile)
identical(x, x2) # returns true

# qs support multithreading
qsave(x, myfile, nthreads=2)
x2 <- qread(myfile, nthreads=2)
identical(x, x2) # returns true

# Other examples
z <- 1:1e7
myfile <- tempfile()
qsave(z, myfile)
z2 <- qread(myfile)
identical(z, z2) # returns true

w <- as.list(rnorm(1e6))
myfile <- tempfile()
qsave(w, myfile)
w2 <- qread(myfile)
identical(w, w2) # returns true
```

`randomStrings` *Generate random strings*

Description

A function for generating a character vector of random strings, for testing purposes.

Usage

```
randomStrings(N, string_size)
```

Arguments

N	The number of random strings to generate
string_size	The number of characters in each string (default 50).

Value

A character vector of random alpha-numeric strings.

Examples

```
randomStrings(N=10, string_size=20) # returns 10 alphanumeric strings of length 20
randomStrings(N=100, string_size=200) # returns 100 alphanumeric strings of length 200
```

---

starnames *Official list of IAU Star Names*

Description

Data from the International Astronomical Union. An official list of the 336 internationally recognized named stars, updated as of June 1, 2018.

Usage

```
data(starnames)
```

Format

A 'data.frame' with official IAU star names and several properties, such as coordinates.

Source

Naming Stars | International Astronomical Union.

References

E Mamajek et. al. (2018), *WG Triennial Report (2015-2018) - Star Names*, Reports on Astronomy, 22 Mar 2018.

Examples

```
data(starnames)
```

zstd_compress_bound *Zstd compress bound*

Description

Exports the compress bound function from the zstd library. Returns the maximum compressed size of an object of length ‘size‘.

Usage

```
zstd_compress_bound(size)
```

Arguments

size	An integer size

Value

maximum compressed size

Examples

```
zstd_compress_bound(100000)
zstd_compress_bound(1e9)
```

zstd_compress_raw *Zstd compression*

Description

Compression of raw vector. Exports the main zstd compression function.

Usage

```
zstd_compress_raw(x, compress_level)
```

Arguments

x	A Raw Vector
compress_level	The compression level (-50 to 22)

Value

The compressed data

Examples

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- zstd_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(zstd_decompress_raw(xcompressed))
```

---

zstd_decompress_raw *Zstd decompression*

Description

Decompresses of raw vector

Usage

```
zstd_decompress_raw(x)
```

Arguments

x	A Raw Vector

Value

The uncompressed data

Examples

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- zstd_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(zstd_decompress_raw(xcompressed))
```

Index