Package ‘qs2’

September 16, 2024

Type Package

Title Efficient Serialization of R Objects

Version 0.1.1

Date 2024-09-16

Maintainer Travers Ching <traversc@gmail.com>

Description Streamlines and accelerates the process of saving and loading R objects, improving speed and compression compared to other methods. The package provides two compression formats: the 'qs2' format, which uses R serialization via the C API while optimizing compression and disk I/O, and the 'qdata' format, featuring custom serialization for slightly faster performance and better compression. Additionally, the 'qs2' format can be directly converted to the standard 'RDS' format, ensuring long-term compatibility with future versions of R.

License GPL-3

Biarch true

Depends R (>= 3.5.0)

Imports Rcpp, stringfish (>= 0.15.1)

LinkingTo Rcpp, stringfish, RcppParallel

Suggests knitr, rmarkdown, dplyr, data.table, stringi

SystemRequirements GNU make

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

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

URL https://github.com/traversc/qs2

BugReports https://github.com/traversc/qs2/issues

NeedsCompilation yes

Author Travers Ching [aut, cre, cph],
Yann Collet [ctb, cph] (Yann Collet is the author of the bundled zstd),
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),
Francesc Alted [ctb, cph] (Shuffling routines derived from Blosc library)

Repository CRAN

Date/Publication 2024-09-16 15:10:06 UTC

Contents

`blosc_shuffle_raw` *Shuffle a raw vector*

Description

Shuffles a raw vector using BLOSC shuffle routines.

Usage

```
blosc_shuffle_raw(data, bytesofsize)
```

Arguments

`data`	A raw vector to be shuffled.
`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

Un-shuffles a raw vector using BLOSC un-shuffle routines.

Usage

```
blosc_unshuffle_raw(data, bytesofsize)
```

Arguments

`data`	A raw vector to be unshuffled.
`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)
```

`qd_read` *qd_read*

Description

Reads an object that was saved to disk in the `qdata` format.

Usage

```
qd_read(file, use_alt_rep = FALSE, validate_checksum=FALSE, nthreads = 1L)
```

Arguments

`file`	The file name/path.
`use_alt_rep`	Use ALTREP when reading in string data (default `FALSE`).
`validate_checksum`	Whether to validate the stored checksum in the file (default `FALSE`). This can be used to test for file corruption but has a performance penality.
`nthreads`	The number of threads to use when reading data (default: `1`).

Value

The object stored in `file`.

Examples

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

# qdata support multithreading
qd_save(x, myfile, nthreads=1)
x2 <- qd_read(myfile, nthreads=1)
identical(x, x2) # returns true
```

qd_save *qd_save*

Description

Saves an object to disk using the `qdata` format.

Usage

```
qd_save(object, file, compress_level = 3L,
shuffle = TRUE, warn_unsupported_types=TRUE,
nthreads = 1L)
```

Arguments

`object`	The object to save.
`file`	The file name/path.
`compress_level`	The compression level used (default 3). The maximum and minimum possible values depends on the version of ZSTD library used. As of ZSTD 1.5.6 the maximum compression level is 22, and the minimum is -131072. Usually, values in the low positive range offer very good performance in terms of speed and compression.
`shuffle`	Whether to allow byte shuffling when compressing data (default: `TRUE`).
`warn_unsupported_types`	Whether to warn when saving an object with an unsupported type (default `TRUE`).
`nthreads`	The number of threads to use when compressing data (default: `1`).

Value

No value is returned. The file is written to disk.

Examples

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

# qdata support multithreading
qd_save(x, myfile, nthreads=1)
x2 <- qd_read(myfile, nthreads=1)
identical(x, x2) # returns true
```

`qs_read` *qs_read*

Description

Reads an object that was saved to disk in the `qs2` format.

Usage

```
qs_read(file, validate_checksum=FALSE, nthreads = 1L)
```

Arguments

`file`	The file name/path.
`validate_checksum`	Whether to validate the stored checksum in the file (default `FALSE`). This can be used to test for file corruption but has a performance penality.
`nthreads`	The number of threads to use when reading data (default: `1`).

Value

The object stored in `file`.

Examples

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

# qs2 support multithreading
qs_save(x, myfile, nthreads=1)
x2 <- qs_read(myfile, nthreads=1)
identical(x, x2) # returns true
```

qs_save *qs_save*

Description

Saves an object to disk using the qs2 format.

Usage

```
qs_save(object, file, compress_level = 3L,
shuffle = TRUE, nthreads = 1L)
```

Arguments

`object`	The object to save.
`file`	The file name/path.
`compress_level`	The compression level used (default 3). The maximum and minimum possible values depends on the version of ZSTD library used. As of ZSTD 1.5.6 the maximum compression level is 22, and the minimum is -131072. Usually, values in the low positive range offer very good performance in terms of speed and compression.
`shuffle`	Whether to allow byte shuffling when compressing data (default: `TRUE`).
`nthreads`	The number of threads to use when compressing data (default: `1`).

Value

No value is returned. The file is written to disk.

Examples

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

# qs2 support multithreading
qs_save(x, myfile, nthreads=1)
x2 <- qs_read(myfile, nthreads=1)
identical(x, x2) # returns true
```

qs_to_rds *qs2 to RDS format*

Description

Converts a file saved in the `qs2` format to the `RDS` format.

Usage

```
qs_to_rds(input_file, output_file, compress_level = 6)
```

Arguments

`input_file`	The `qs2` file to convert.
`output_file`	The `RDS` file to write.
`compress_level`	The gzip compression level to use when writing the RDS file (a value between 0 and 9).

Value

No value is returned. The converted file is written to disk.

Examples

```
qs_tmp <- tempfile(fileext = ".qs2")
rds_tmp <- tempfile(fileext = ".RDS")

x <- runif(1e6)
qs_save(x, qs_tmp)
qs_to_rds(input_file = qs_tmp, output_file = rds_tmp)
x2 <- readRDS(rds_tmp)
stopifnot(identical(x, x2))
```

qx_dump *qx_dump*

Description

Exports the uncompressed binary serialization to a list of raw vectors for both `qs2` and `qdata` formats. For testing and exploratory purposes mainly.

Usage

```
qx_dump(file)
```

Arguments

`file`	A file name/path.

Value

The uncompressed serialization.

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
        num = rnorm(1e3),
        char = sample(state.name, 1e3, replace=TRUE),
        stringsAsFactors = FALSE)
myfile <- tempfile()
qs_save(x, myfile)
binary_data <- qx_dump(myfile)
```

`rds_to_qs` *RDS to qs2 format*

Description

Converts a file saved in the `RDS` format to the `qs2` format.

Usage

```
rds_to_qs(input_file, output_file, compress_level = 3)
```

Arguments

`input_file`	The `RDS` file to convert.
`output_file`	The `qs2` file to write.
`compress_level`	The zstd compression level to use when writing the `qs2` file. See the `qs_save` help file for more details on this parameter.

Details

The `shuffle` parameters is currently not supported when converting from `RDS` to `qs2`. When reading the resulting `qs2` file, `validate_checksum` must be set to `FALSE`.

Value

No value is returned. The converted file is written to disk.

Examples

```
qs_tmp <- tempfile(fileext = ".qs2")
rds_tmp <- tempfile(fileext = ".RDS")

x <- runif(1e6)
saveRDS(x, rds_tmp)
rds_to_qs(input_file = rds_tmp, output_file = qs_tmp)
x2 <- qs_read(qs_tmp, validate_checksum = FALSE)
stopifnot(identical(x, x2))
```

`zstd_compress_bound` *Zstd compress bound*

Description

Exports the compress bound function from the zstd library. Returns the maximum potential 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

Compresses to a raw vector using the zstd algorithm. Exports the main zstd compression function.

Usage

```
zstd_compress_raw(data, compress_level)
```

Arguments

`data`	Raw vector to be compressed.
`compress_level`	The compression level used.

Value

The compressed data as a raw vector.

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 a zstd compressed raw vector.

Usage

```
zstd_decompress_raw(data)
```

Arguments

`data`	A raw vector to be decompressed.

Value

The decompressed data as a raw vector.

Examples

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

Index