Package: qs2 (via r-universe)

November 25, 2024

Type Package

Title Efficient Serialization of R Objects

Version 0.1.2

Date 2024-11-23

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

LazyData true

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/qsbase/qs2

BugReports https://github.com/qsbase/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 2024-11-24 05:30:02 UTC

Config/pak/sysreqs make

Contents

base85_decode *Z85 Decoding*

Description

Decodes a Z85 encoded string back to binary

Usage

```
base85_decode(encoded_string)
```

Arguments

encoded_string	A string.

Value

The original raw vector.

base85_encode *Z85 Encoding*

Description

Encodes binary data (a raw vector) as ASCII text using Z85 encoding format.

Usage

```
base85_encode(rawdata)
```

Arguments

rawdata	A raw vector.

Details

Z85 is a binary to ASCII encoding format created by Pieter Hintjens in 2010 and is part of the ZeroMQ RFC. The encoding has a dictionary using 85 out of 94 printable ASCII characters. There are other base 85 encoding schemes, including Ascii85, which is popularized and used by Adobe. Z85 is distinguished by its choice of dictionary, which is suitable for easier inclusion into source code for many programming languages. The dictionary excludes all quote marks and other control characters, and requires no special treatment in R and most other languages. Note: although the official specification restricts input length to multiples of four bytes, the implementation here works with any input length. The overhead (extra bytes used relative to binary) is 25%. In comparison, base 64 encoding has an overhead of 33.33%.

Value

A string representation of the raw vector.

References

https://rfc.zeromq.org/spec/32/

`base91_decode` *basE91 Decoding*

Description

Decodes a basE91 encoded string back to binary

Usage

```
base91_decode(encoded_string)
```

Arguments

`encoded_string`	A string.

Value

The original raw vector.

`base91_encode` *basE91 Encoding*

Description

Encodes binary data (a raw vector) as ASCII text using basE91 encoding format.

Usage

```
base91_encode(rawdata, quote_character = "\"")
```

Arguments

`rawdata`	A raw vector.
`quote_character`	The character to use in the encoding, replacing the double quote character. Must be either a single quote (`"'"`), a double quote (`"\""`) or a dash (`"-"`).

Details

basE91 (capital E for stylization) is a binary to ASCII encoding format created by Joachim Henke in 2005. The overhead (extra bytes used relative to binary) is 22.97% on average. In comparison, base 64 encoding has an overhead of 33.33%. The original encoding uses a dictionary of 91 out of 94 printable ASCII characters excluding - (dash), \ (backslash) and ' (single quote). The original encoding does include double quote characters, which are less than ideal for strings in R. Therefore, you can use the `quote_character` parameter to substitute dash or single quote.

Value

A string representation of the raw vector.

References

https://base91.sourceforge.net/

`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)
```

catquo *catquo*

Description

Prints a string with single quotes on a new line.

Usage

```
catquo(...)
```

Arguments

`...`	Arguments passed on to `cat()`.

decode_source *Decode a compressed string*

Description

A helper function for encoding and compressing a file or string to ASCII using `base91_encode()` and `qs_serialize()` with the highest compression level.

Usage

```
decode_source(string)
```

Arguments

`string`	A string to decode.

Value

The original (decoded) object.

See Also

`encode_source()` for more details.

encode_source *Encode and compress a file or string*

Description

A helper function for encoding and compressing a file or string to ASCII using `base91_encode()` and `qs_serialize()` with the highest compression level.

Usage

```
encode_source(x = NULL, file = NULL, width = 120)
```

Arguments

`x`	The object to encode (if `file` is not NULL)
`file`	The file to encode (if `x` is not NULL)
`width`	The output will be broken up into individual strings, with `width` being the longest allowable string.

Details

The encode_source() and decode_source() functions are useful for storing small amounts of data or text inline to a .R or .Rmd file.

Value

A character vector in base91 representing the compressed original file or object.

Examples

```
set.seed(1); data <- sample(500)
result <- encode_source(data)
# Note: the result string is not guaranteed to be consistent between qs or zstd versions
#       but will always properly decode regardless
print(result)
result <- decode_source(result) # [1]  1  2  3  4  5  6  7  8  9 10
```

---

qd_deserialize *qd_deserialize*

Description

Deserializes a raw vector to an object using the qdata format.

Usage

```
qd_deserialize(input, use_alt_rep = FALSE, validate_checksum = FALSE, nthreads = 1L)
```

Arguments

input	The raw vector to deserialize.
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 deserialized object.

Examples

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

---

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
```

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
```

qd_serialize *qd_serialize*

Description

Serializes an object to a raw vector using the qdata format.

Usage

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

Arguments

object	The object to save.
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

The serialized object as a raw vector.

Examples

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

qs_deserialize *qs_deserialize*

Description

Deserializes a raw vector to an object using the qs2 format.

Usage

```
qs_deserialize(input, validate_checksum = FALSE, nthreads = 1L)
```

Arguments

input	The raw vector to deserialize.
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 deserialized object.

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
        num = rnorm(1e3),
        char = sample(state.name, 1e3, replace=TRUE),
        stringsAsFactors = FALSE)
xserialized <- qs_serialize(x)
x2 <- qs_deserialize(xserialized)
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
```

---

qs_readm *qs_readm*

Description

Reads an object in a file serialized to disk using qs_savem().

Usage

```
qs_readm(file, env = parent.frame(), ...)
```

Arguments

file	The file name/path.
env	The environment where the data should be loaded. Default is the calling environment (parent.frame()).
...	additional arguments will be passed to qs_read.

Details

This function extends qs_read to replicate the functionality of base::load() to load multiple saved objects into your workspace.

Value

Nothing is explicitly returned, but the function will load the saved objects into the workspace.

Examples

```
x1 <- data.frame(int = sample(1e3, replace=TRUE),
                 num = rnorm(1e3),
                 char = sample(starnames$`IAU Name`, 1e3, replace=TRUE),
                 stringsAsFactors = FALSE)
x2 <- data.frame(int = sample(1e3, replace=TRUE),
                 num = rnorm(1e3),
                 char = sample(starnames$`IAU Name`, 1e3, replace=TRUE),
                 stringsAsFactors = FALSE)
myfile <- tempfile()
qs_savem(x1, x2, file=myfile)
rm(x1, x2)
qs_readm(myfile)
exists('x1') && exists('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
```

`qs_savem` *qs_savem*

Description

Saves (serializes) multiple objects to disk.

Usage

```
qs_savem(...)
```

Arguments

`...`	Objects to serialize. Named arguments will be passed to `qs_save()` during saving. Un-named arguments will be saved. A named `file` argument is required.

Details

This function extends `qs_save()` to replicate the functionality of `base::save()` to save multiple objects. Read them back with `qs_readm()`.

Examples

```
x1 <- data.frame(int = sample(1e3, replace=TRUE),
                 num = rnorm(1e3),
                 char = sample(starnames$`IAU Name`, 1e3, replace=TRUE),
                 stringsAsFactors = FALSE)
x2 <- data.frame(int = sample(1e3, replace=TRUE),
                 num = rnorm(1e3),
                 char = sample(starnames$`IAU Name`, 1e3, replace=TRUE),
                 stringsAsFactors = FALSE)
myfile <- tempfile()
qs_savem(x1, x2, file=myfile)
rm(x1, x2)
qs_readm(myfile)
exists('x1') && exists('x2') # returns true
```

qs_serialize *qs_serialize*

Description

Serializes an object to a raw vector using the qs2 format.

Usage

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

Arguments

object	The object to save.
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

The serialized object as a raw vector.

Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
        num = rnorm(1e3),
        char = sample(state.name, 1e3, replace=TRUE),
        stringsAsFactors = FALSE)
xserialized <- qs_serialize(x)
x2 <- qs_deserialize(xserialized)
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))
```

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)
```

xxhash_raw *XXH3_64 hash*

Description

Calculates 64-bit XXH3 hash

Usage

```
xxhash_raw(data)
```

Arguments

`data`	The data to hash

Value

The 64-bit hash

Examples

```
x <- as.raw(c(1,2,3))
xxhash_raw(x)
```

`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