Using the qs package

qs – quick serialization of R objects

Inspired by the `fst` package, this package aims to provide functions for quickly writing (serialization) and reading (de-serialization) R objects to and from disk. In comparison, this package takes a more general approach for attributes and object references, allowing for serialization of various data structures, such as lists, nested lists and attributes Because of this more general approach, any S3 object built on common data types can also be serialized. E.g., `tibbles`, time-stamps, `bit64`, etc.

Features

The table below compares the features of different serialization approaches in R.

	qs	fst	saveRDS
Not Slow	Yes	Yes	No
Numeric Vectors	Yes	Yes	Yes
Integer Vectors	Yes	Yes	Yes
Logical Vectors	Yes	Yes	Yes
Character Vectors	Yes	Yes	Yes
Character Encoding	Yes	(vector-wide only)	Yes
Complex Vectors	Yes	No	Yes
Data.Frames	Yes	Yes	Yes
On disk row access	No	Yes	No
Attributes	Yes	Some	Yes
Lists / Nested Lists	Yes	No	Yes
Multi-threaded	No (Not Yet)	Yes	No

Installation:

1. `devtools::install_github("traversc/qs")`

Example:

See `tests/correctness_testing.r` for more examples. Below is an example serializing a large `data.frame` to disk.

```r
library(qs)
x1 <- data.frame(int = sample(5e6, replace=T),
                 num = rnorm(5e6),
                 char = randomStrings(5e6), stringsAsFactors = F)
qsave(x1, "/tmp/mydata.qs")

x2 <- qread("/tmp/mydata.qs")
identical(x1, x2) # returns true
```

```
## [1] TRUE
```

Benchmarks

Data.Frame benchmark

Benchmarks for serializing and de-serializing large data.frames (5 million rows) composed of a numeric column (`rnorm`), an integer column (`sample(5e6)`), and a character vector column (random alphanumeric strings of length 50). See `vignettes/dataframe_bench.png` for a comparison using different compression parameters.

Serialization speed with default parameters:

Method	write time (s)	read time (s)
qs	0.49391294	8.8818166
fst (1 thread)	0.37411811	8.9309314
fst (4 thread)	0.3676273	8.8565951
saveRDS	14.377122	12.467517

Serialization speed with different parameters

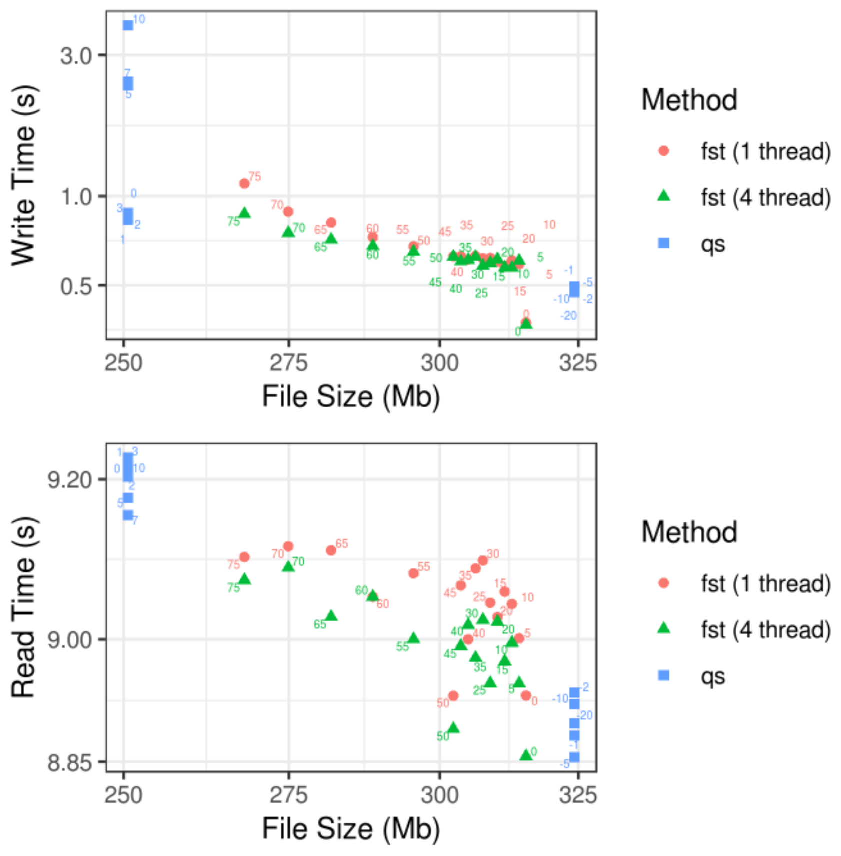

Figure 1: dataframe_bench

Nested List benchmark

Benchmarks for serialization of random nested lists with random attributes (approximately 50 Mb). See the nested list example in the tests/correctness_testing.r.

Serialization speed with default parameters

Method	write time (s)	read time (s)
qs	0.17840716	0.19489372
saveRDS	3.484225	0.58762548

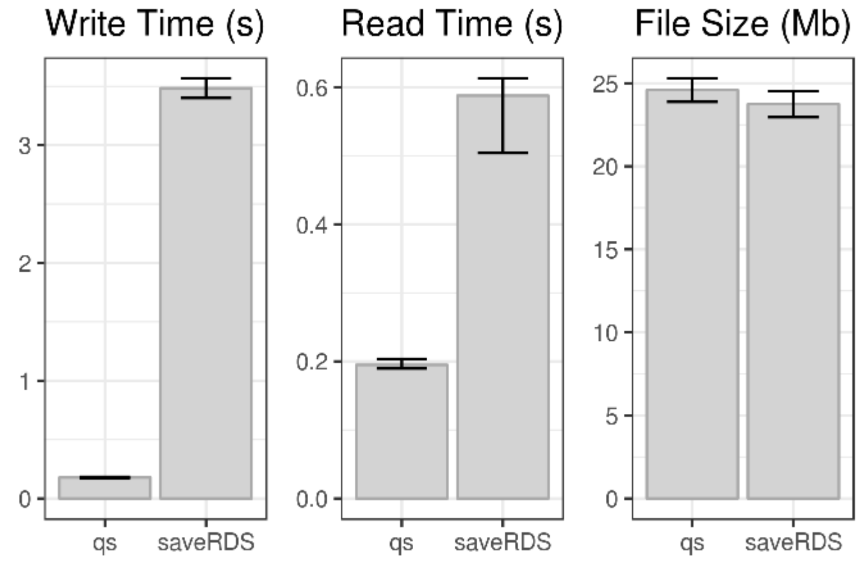

Figure 2: nested_list_bench