Fast, Faster, ???

Comparing Data Read/Write Speeds in R

David Zimmermann

R User Group Cologne

2020-03-05 @ Cologne Intelligence

How can we write and read data in R efficiently?

Or

WHAT? ... you still use read.csv?

Soooo many options...

Filetypes

- `.csv`
- `.rds`
- `.fst`
- `.qs`
- `.feather`
- `.arrow`
- `.parquet`

+ different compression levels

R packages

- `base`
- `readr`
- `data.table`
- `vroom`
- `fst`
- `qs`
- `feather`
- `arrow`

Multitude of wonderful packages

	Filetypes						
Library	.csv	.rds	.fst	.qs	.feather	.arrow	.parquet
base	✓	✓					
readr	✓	✓					
data.table	✓						
vroom	✓						
fst			✓				
qs				✓			
feather					✓		
arrow					✓	✓	✓

And the winner is?

qs

... but, its more complicated

The many factors to consider

Factors to consider

- RAM usage
- Read/Write speed
- Filesize

depends on ...

Type of data

- size (KBs, MBs, GBs)
- shape (long, wide)
- type (string, int, numeric, lists in dataframes)

+

Additional factors

- Python compatiblity/file formats
- Dependency weight
- Partial read (row / column wise)
- Support of nested data-frames

Just give me one option to use!

The Real Winner is

I want to ...

- have the fastest read/writes: **`fst`**
- have the smallest RAM usage: **`qs`**
- have the smallest filesize on disk: **`qs`** or **`base->rds`** (slow...)
- communicate with everything: **`data.table`**
- have light dependencies: **`base`** or **`data.table`**
- also use it in python: **`arrow->feather`**
- use nested dataframes: **`qs`**

Source code can be found here: https://github.com/DavZim/io-benchmarks

Summary

		Capabilities					Factors		
Library	Fileformat	Multicore	Nested DFs	Python	Random Access	Dependencies	Speed	RAM	Filesize
Plain Text									
base	.csv			✓		👑	👎	👉	👉
readr	.csv			✓		👉	👎	👎	👎
data.table	.csv	✓		✓	⌄	👑	👎	👉	👎
vroom	.csv	✓		✓	⌄	👉			
R specific									
base	.rds		✓			👑	👉	👍	👍
readr	.rds		✓			👉	👍	👍	👍
fst	.fst	✓		?	👑	👍	👍	👉	👉
qs	.qs	✓	✓			👍	👑	👑	👑
Arrow-based									
feather	.feather			✓		👍	👉	👉	👉
arrow	.feather			✓	⌄	👉	👉	👎	👉
arrow	.arrow			✓		👉	👉	👎	👎
arrow	.parquet			✓	⌄	👉	👎	👎	👎

Note that `vroom` is not properly benchmarked, due to lazy loading(?!)

Special Mentions & Remarks

Special Mentions

- `data.table` for its use of pipes: (search all `.csv.gz` archives for the line `rug-meetup` and load only those lines)

```
fread(cmd = "cat *.csv.gz | gunzip | grep -e rug-meetup")
```

- `vroom` for its excellent functionality and performance with strings
- `qs` for its excellent compression, i.e., `preset="archive"` reduces the `tickets` dataset from 3.4 GB in `R` to 278 MB (!) on disk in ~3 mins or in 12 secs to 580 MB with `preset="balanced"`

```
qsave(tickets, "tickets.qs", preset = "archive")
```

Remarks

- `vroom` loads the data only lazily, resulting in inaccurate measurements (i.e., 2 MB when GBs where used on the machine), it is a lot better than these benchmarks suggest.
- `qs` allows multicore, but default is to only use 1, for using all available cores, use

```
qs::qread("myfile.qs", nthrads = parallel::detectCores())
```

Thats it for today

... OK, ok, BUT I still want the details...

see following slides

Average Speed

		Read					Write				
		tiny	wide	large	flights	tickets	tiny	wide	large	flights	tickets
Library	Fileformat	3.4 kB	931 kB	400 MB	41 MB	3.4 GB	3.4 kB	931 kB	400 MB	41 MB	3.4 GB
Plain Text											
base	.csv	1ms	87ms	1m 7.9s	1.3s	1m 38.9s	1ms	97ms	40.4s	3.5s	2m 7.2s
readr	.csv	2ms	126ms	18.4s	496ms	37.3s	2ms	83ms	9.3s	972ms	1m 22.4s
data.table	.csv	1ms	30ms	347ms	68ms	14.3s	1ms	30ms	1.8s	132ms	4.5s
vroom	.csv	NA	NA	NA	NA	NA	3ms	52ms	3s	374ms	14.3s
R specific											
base	.rds	1ms	6ms	2s	345ms	54.7s	1ms	32ms	15.6s	1.6s	1m 46.8s
readr	.rds	1ms	2ms	399ms	220ms	43.4s	1ms	3ms	1.3s	322ms	33.5s
fst	.fst	1ms	4ms	80ms	53ms	13.4s	1ms	6ms	1s	69ms	7s
qs	.qs	1ms	2ms	258ms	78ms	15.5s	1ms	8ms	1.1s	130ms	16s
Arrow-based											
feather	.feather	1ms	11ms	78ms	47ms	12.1s	1ms	4ms	1.3s	153ms	10.5s
arrow	.feather	2ms	5ms	59ms	52ms	13.3s	2ms	13ms	992ms	85ms	10.2s
arrow	.arrow	2ms	5ms	101ms	58ms	14.5s	2ms	15ms	1.3s	155ms	14s
arrow	.parquet	2ms	20ms	114ms	64ms	14.4s	3ms	41ms	1.3s	141ms	15.1s

Average RAM Usage

		Read					Write				
		tiny	wide	large	flights	tickets	tiny	wide	large	flights	tickets
Library	Fileformat	3.4 kB	931 kB	400 MB	41 MB	3.4 GB	3.4 kB	931 kB	400 MB	41 MB	3.4 GB
Plain Text											
base	.csv	359 kB	12 MB	2.2 GB	252 MB	19 GB	144 kB	12 MB	16 MB	145 MB	423 MB
readr	.csv	1.3 MB	22 MB	442 MB	59 MB	4.1 GB	96 kB	12 MB	136 kB	30 MB	195 MB
data.table	.csv	400 kB	1.3 MB	440 MB	36 MB	3.7 GB	130 kB	139 kB	130 kB	130 kB	130 kB
vroom	.csv	NA kB	NA kB	NA kB	NA kB	NA kB	512 kB	13 MB	555 kB	30 MB	195 MB
R specific											
base	.rds	1.1 kB	865 kB	400 MB	40 MB	3.2 GB	8.8 kB	8.8 kB	8.8 kB	8.8 kB	8.8 kB
readr	.rds	3.5 kB	868 kB	400 MB	40 MB	3.2 GB	19 kB	19 kB	19 kB	19 kB	19 kB
fst	.fst	79 kB	951 kB	404 MB	42 MB	3.3 GB	381 kB	381 kB	382 kB	381 kB	382 kB
qs	.qs	0.0 kB	864 kB	400 MB	40 MB	3.2 GB	0.0 kB	0.0 kB	0.0 kB	0.0 kB	10.0 kB
Arrow-based											
feather	.feather	68 kB	1.2 MB	400 MB	40 MB	3.2 GB	35 kB	35 kB	35 kB	35 kB	35 kB
arrow	.feather	3.3 MB	4.1 MB	403 MB	44 MB	3.2 GB	3.4 MB	4.2 MB	403 MB	30 MB	754 MB
arrow	.arrow	2.7 MB	3.5 MB	403 MB	43 MB	3.2 GB	5.0 MB	5.8 MB	405 MB	32 MB	755 MB
arrow	.parquet	3.2 MB	4.1 MB	403 MB	44 MB	3.2 GB	5.0 MB	5.8 MB	405 MB	32 MB	755 MB

Average Filesize

Library	Fileformat	tiny 3.4 kB	wide 931 kB	large 400 MB	flights 41 MB	tickets 3.4 GB
Plain Text						
`base`	.csv	2.0 kB	1.8 MB	917 MB	36 MB	3.0 GB
`readr`	.csv	2.1 kB	2.0 MB	982 MB	32 MB	2.3 GB
`data.table`	.csv	1.9 kB	1.8 MB	908 MB	31 MB	2.3 GB
`vroom`	.csv	2.1 kB	2.0 MB	982 MB	32 MB	2.3 GB
R specific						
`base`	.rds	1.0 kB	773 kB	384 MB	7.0 MB	362 MB
`readr`	.rds	1.2 kB	827 kB	400 MB	45 MB	4.3 GB
`fst`	.fst	1.7 kB	864 kB	402 MB	16 MB	1.4 GB
`qs`	.qs	0.9 kB	735 kB	353 MB	5.7 MB	342 MB
Arrow-based						
`feather`	.feather	1.7 kB	876 kB	400 MB	40 MB	3.0 GB
`arrow`	.feather	1.7 kB	876 kB	400 MB	40 MB	3.0 GB
`arrow`	.arrow	2.6 kB	952 kB	400 MB	40 MB	3.0 GB
`arrow`	.parquet	3.7 kB	1.2 MB	414 MB	5.9 MB	642 MB

Note that all comparisons are performed without compression

What about compressions?

Average Speed Compression

			Read					Write				
			tiny	wide	large	flights	tickets	tiny	wide	large	flights	tickets
Library	Fileformat	Compression	3.4 kB	931 kB	400 MB	41 MB	3.4 GB	3.4 kB	931 kB	400 MB	41 MB	3.4 GB
base												
base	.rds	bzip2	1ms	6ms	1.9s	340ms	57.3s	1ms	82ms	38.5s	4.1s	10m 39.6s
base	.rds	gzip	2ms	6ms	2.1s	341ms	55.2s	1ms	34ms	14.6s	1.7s	2m 1.5s
base	.rds	xz	1ms	6ms	2.1s	337ms	54.6s	2ms	135ms	2m 3.3s	15s	13m 48.5s
readr												
readr	.rds	bz2	1ms	2ms	397ms	237ms	45.8s	1ms	89ms	38.7s	4.2s	11m 42s
readr	.rds	bzip2	1ms	2ms	375ms	215ms	43.5s	NA	NA	NA	NA	NA
readr	.rds	gz	1ms	2ms	421ms	231ms	44.4s	1ms	33ms	15s	1.7s	1m 52.1s
readr	.rds	xz	1ms	2ms	379ms	230ms	44.3s	2ms	141ms	2m 2.6s	14.6s	14m 7.3s
fst												
fst	.fst	0	1ms	4ms	80ms	54ms	12.6s	1ms	3ms	1.1s	116ms	8.5s
fst	.fst	100	1ms	4ms	80ms	53ms	11.9s	1ms	29ms	2.8s	589ms	27.4s
qs												
qs	.qs	archive	1ms	2ms	254ms	78ms	15.3s	18ms	70ms	13.9s	1.7s	2m 51s
qs	.qs	balanced	1ms	2ms	262ms	76ms	14.8s	1ms	3ms	1s	94ms	12.5s
qs	.qs	fast	1ms	1ms	258ms	75ms	15.1s	1ms	2ms	654ms	106ms	12s

Average RAM Usage Compression

			Read					Write				
			tiny	wide	large	flights	tickets	tiny	wide	large	flights	tickets
Library	Fileformat	Compression	3.4 kB	931 kB	400 MB	41 MB	3.4 GB	3.4 kB	931 kB	400 MB	41 MB	3.4 GB
base												
base	.rds	bzip2	1.1 kB	865 kB	400 MB	40 MB	3.2 GB	12 kB	12 kB	12 kB	12 kB	12 kB
base	.rds	gzip	1.1 kB	865 kB	400 MB	40 MB	3.2 GB	8.8 kB	8.8 kB	8.8 kB	8.8 kB	8.8 kB
base	.rds	xz	1.1 kB	865 kB	400 MB	40 MB	3.2 GB	12 kB	12 kB	12 kB	12 kB	12 kB
readr												
readr	.rds	bz2	3.5 kB	868 kB	400 MB	40 MB	3.2 GB	22 kB	22 kB	22 kB	22 kB	22 kB
readr	.rds	bzip2	3.5 kB	868 kB	400 MB	40 MB	3.2 GB	NA kB	NA kB	NA kB	NA kB	NA kB
readr	.rds	gz	3.5 kB	868 kB	400 MB	40 MB	3.2 GB	19 kB	19 kB	19 kB	19 kB	19 kB
readr	.rds	xz	3.5 kB	868 kB	400 MB	40 MB	3.2 GB	22 kB	22 kB	22 kB	22 kB	22 kB
fst												
fst	.fst	0	79 kB	951 kB	404 MB	42 MB	3.3 GB	381 kB	381 kB	382 kB	381 kB	382 kB
fst	.fst	100	79 kB	951 kB	404 MB	42 MB	3.3 GB	381 kB	381 kB	382 kB	381 kB	382 kB
qs												
qs	.qs	archive	0.0 kB	864 kB	400 MB	40 MB	3.2 GB	0.0 kB	0.0 kB	0.0 kB	0.0 kB	10.0 kB
qs	.qs	balanced	0.0 kB	864 kB	400 MB	40 MB	3.2 GB	0.0 kB	0.0 kB	0.0 kB	0.0 kB	10.0 kB
qs	.qs	fast	0.0 kB	864 kB	400 MB	40 MB	3.2 GB	0.0 kB	0.0 kB	0.0 kB	0.0 kB	10.0 kB

Average Filesize Compression

Library	Fileformat	Compression	tiny 3.4 kB	wide 931 kB	large 400 MB	flights 41 MB	tickets 3.4 GB
base							
base	.rds	bzip2	1.2 kB	779 kB	387 MB	4.3 MB	251 MB
base	.rds	gzip	1.0 kB	773 kB	384 MB	7.0 MB	362 MB
base	.rds	xz	1.0 kB	759 kB	372 MB	4.7 MB	252 MB
readr							
readr	.rds	bz2	1.2 kB	779 kB	387 MB	4.3 MB	251 MB
readr	.rds	bzip2	NA kB	NA kB	NA kB	NA kB	NA kB
readr	.rds	gz	1.0 kB	773 kB	384 MB	7.0 MB	362 MB
readr	.rds	xz	1.0 kB	759 kB	372 MB	4.7 MB	252 MB
fst							
fst	.fst	0	1.5 kB	843 kB	400 MB	40 MB	3.1 GB
fst	.fst	100	1.8 kB	869 kB	384 MB	7.3 MB	564 MB
qs							
qs	.qs	archive	0.9 kB	732 kB	351 MB	5.3 MB	278 MB
qs	.qs	balanced	0.9 kB	764 kB	377 MB	8.8 MB	580 MB
qs	.qs	fast	1.0 kB	811 kB	402 MB	15 MB	905 MB

Code Examples

Code Examples

Plain text file formats

Package `base`: `write.csv()` & `read.csv()`

```
write.csv(df, "dataset.csv")
df2 <- read.csv("dataset.csv")
```

Package `readr`: `write_csv()` & `read_csv()`

```
library(readr)
write_csv(df, "dataset.csv")
df2 <- read_csv("dataset.csv")
```

Package `data.table`: `fwrite()` & `fread()`

```
library(data.table)
fwrite(df, "dataset.csv")
df2 <- fread("dataset.csv")
```

Package `vroom`: `vroom_write()` & `vroom()`

```
library(vroom)
vroom_write(df, "dataset.csv")
df2 <- vroom("dataset.csv")
```

R specific file formats

Package `base`: `saveRDS()` & `readRDS()`

```
saveRDS(df, "dataset.rds", compress = "none")
df2 <- readRDS("dataset.rds")
```

Package `readr`: `write_rds()` & `read_rds()`

```
library(readr)
write_rds(df, "dataset.rds", compress = "none")
df2 <- read_rds("dataset.rds")
```

Package `fst`: `fst_write()` & `fst_read()`

```
library(fst)
write_fst(df, "dataset.fst", compression = 50)
df2 <- read_fst("dataset.fst")
```

Package `qs`: `qsave()` & `qread()`

```
library(qs)
qsave(df, "dataset.qs", preset = "high")
df2 <- qread("dataset.qs")
```

Code Examples cont'd

arrow based file formats

Package `feather`: `write_feather()` & `read_feather()`

```
library(feather)
write_feather(df, "dataset.feather")
df2 <- read_feather("dataset.feather")
```

Package `arrow`: `write_feather()` & `read_feather()`

```
library(arrow)
# install_arrow()
write_feather(df, "dataset.feather")
df2 <- read_feather("dataset.feather")
```

Package `arrow`: `write_arrow()` & `read_arrow()`

```
library(arrow)
# install_arrow()
write_arrow(df, "dataset.arrow")
df2 <- read_arrow("dataset.arrow")
```

Package `arrow`: `write_parquet()` & `read_parquet()`

```
library(arrow)
# install_arrow()
write_parquet(df, "dataset.parquet")
df2 < read_parquet("dataset.parquet")
```

System Information

Hardware

- Dell XPS 15 Laptop
- CPU: Intel Core i7 (8th gen)
- Model: 9750H
- Clock speed: 2.60GHz
- Threads: 6 cores, 12 threads
- RAM: 32 GB
- SSD with ~900 MB/sec read speed

Software Platform

```
sessioninfo::platform_info()
```

```
##  setting  value
##  version  R version 3.6.2 (2019-12-12)
##  os       Ubuntu 18.04.4 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language en
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       Europe/Berlin
##  date     2020-03-05
```

Benchmark package versions

```
pkgs <-  c("readr", "data.table", "vroom", "fst", "qs", "feather", "arrow")
sessioninfo::package_info(pkgs)
```

```
##  package       * version  date       lib source
##  arrow           0.15.1.1 2019-11-05 [1] CRAN (R 3.6.2)
##  assertthat      0.2.1    2019-03-21 [1] CRAN (R 3.6.2)
##  BH              1.72.0-3 2020-01-08 [1] CRAN (R 3.6.2)
##  bit             1.1-15.2 2020-02-10 [1] CRAN (R 3.6.2)
##  bit64           0.9-7    2017-05-08 [1] CRAN (R 3.6.2)
##  cli             2.0.1    2020-01-08 [1] CRAN (R 3.6.2)
##  clipr           0.7.0    2019-07-23 [1] CRAN (R 3.6.2)
##  crayon          1.3.4    2017-09-16 [1] CRAN (R 3.6.2)
##  data.table      1.12.8   2019-12-09 [1] CRAN (R 3.6.2)
##  digest          0.6.24   2020-02-12 [1] CRAN (R 3.6.2)
##  ellipsis        0.3.0    2019-09-20 [1] CRAN (R 3.6.2)
##  fansi           0.4.1    2020-01-08 [1] CRAN (R 3.6.2)
##  feather         0.3.5    2019-09-15 [1] CRAN (R 3.6.2)
##  fst             0.9.0    2019-04-09 [1] CRAN (R 3.6.2)
##  glue            1.3.1    2019-03-12 [1] CRAN (R 3.6.2)
##  hms             0.5.3    2020-01-08 [1] CRAN (R 3.6.2)
##  lifecycle       0.1.0    2019-08-01 [1] CRAN (R 3.6.2)
##  magrittr        1.5      2014-11-22 [1] CRAN (R 3.6.2)
##  pillar          1.4.3    2019-12-20 [1] CRAN (R 3.6.2)
##  pkgconfig       2.0.3    2019-09-22 [1] CRAN (R 3.6.2)
##  prettyunits     1.1.1    2020-01-24 [1] CRAN (R 3.6.2)
##  progress        1.2.2    2019-05-16 [1] CRAN (R 3.6.2)
##  purrr         * 0.3.3    2019-10-18 [1] CRAN (R 3.6.2)
##  qs              0.21.1   2020-02-07 [1] CRAN (R 3.6.2)
##  R6              2.4.1    2019-11-12 [1] CRAN (R 3.6.2)
##  RApiSerialize   0.1.0    2014-04-19 [1] CRAN (R 3.6.2)
##  Rcpp            1.0.3    2019-11-08 [1] CRAN (R 3.6.2)
##  readr         * 1.3.1    2018-12-21 [1] CRAN (R 3.6.2)
##  rlang           0.4.4    2020-01-28 [1] CRAN (R 3.6.2)
##  tibble        * 2.1.3    2019-06-06 [1] CRAN (R 3.6.2)
##  tidyselect      1.0.0    2020-01-27 [1] CRAN (R 3.6.2)
##  utf8            1.1.4    2018-05-24 [1] CRAN (R 3.6.2)
##  vctrs           0.2.2    2020-01-24 [1] CRAN (R 3.6.2)
##  vroom           1.2.0    2020-01-13 [1] CRAN (R 3.6.2)
##  withr           2.1.2    2018-03-15 [1] CRAN (R 3.6.2)
##
## [1] /home/CO-IN.LOCAL/dzimmermann/R/x86_64-pc-linux-gnu-library/3.6
## [2] /usr/local/lib/R/site-library
## [3] /usr/lib/R/site-library
## [4] /usr/lib/R/library
```