# Biggish Data: Tips and tricks for working with kinda big data in R

Jeff Hollister

repo: https://github.com/usepa/biggish_data_r

(requires GHE license)

2024-04-17

Corvallis Spatial Huddle

# Biggish Data

- Most data not “Big”
- Lot of data “biggish”
  - Storage/Sharing challenges
  - Read/Write challenges
  - Analysis challenges
- Useful for all datasets
  - Less storage
  - Less CPU time
  - Less bandwidth
  - The “cloud” isn’t free.

# Outline

- The example
- Writing data
- Reading data
- File sizes
- Summarizing data
- S3

# The example - Biggish National Lakes Assessment 2017

```
library(dplyr)
library(readr)
url <- "https://www.epa.gov/sites/default/files/2021-
04/nla_2017_water_chemistry_chla-data.csv"
nla17 <- read_csv(url)
nla_big <-
nla17[sample(1:nrow(nla17),10000000,replace=TRUE),]
nla_big <- rename(nla_big, state = STATE,
                  analyte = ANALYTE,
                  result = RESULT)
```

# The example - Biggish National Lakes Assessment 2017

```
dim(nla_big)
[1] 10000000       23
format(object.size(nla_big), "Gb")
[1] "1.7 Gb"
```

# Packages

```
library(data.table)
library(readr)
library(arrow)
library(tictoc) #Timing
```

- Apache Arrow
- Language independent file format(s) for columnar datasets
- parquet, geoparquet, feather, Arrow csv

# Writing data

- utils::write.csv (.csv)
- readr::write_csv (.csv)
- data.table::fwrite (.csv)
- arrow::write_csv_arrow (.csv)
- base::save (.rda)
- arrow::write_feather (.feather)
- arrow::write_parquet (.parquet)
- arrow::write_dataset (partions/multiple .parquet)

# Writing data - `utils::write.csv`

```
tic()
write.csv(nla_big, "../data/nla_big.csv")
toc()
```

```
539.04 sec elapsed
```

# Writing data - `readr::write_csv`

```
tic()
write_csv(nla_big, "../data/nla_big.csv")
toc()
```

```
6.58 sec elapsed
```

# Writing data - `data.table::fwrite`

```
tic()
fwrite(nla_big, "../data/nla_big.csv")
toc()
```

```
6.25 sec elapsed
```

# Writing data - `arrow::write_csv_arrow`

```
tic()
write_csv_arrow(nla_big, file = "../data/nla_big.csv")
toc()
```

```
23.28 sec elapsed
```

# Writing data - `base::save`

```
tic()
save(nla_big, file = "../data/nla_big.rda")
toc()
```

```
159.36 sec elapsed
```

# Writing data - `arrow::write_feather`

```
tic()
write_feather(nla_big, sink = "../data/nla_big.feather",
              compression = "zstd")
toc()
```

```
10.02 sec elapsed
```

# Writing data - `arrow::write_parquet`

```
tic()
write_parquet(nla_big, sink = "../data/nla_big.parquet",
              compression = "zstd")
toc()
```

```
22.45 sec elapsed
```

# Writing data - `arrow::write_dataset` multiple file format

```
tic()
nla_big |>
  group_by(state) |>
  write_dataset(path = "../data/nla_big", compression =
"zstd")
toc()
```

```
23.95 sec elapsed
```

# Writing data - Times

| function | time |
|---|---|
| utils::write.csv | 539.0 [s] |
| base::save | 159.4 [s] |
| arrow::write_parquet partition | 23.9 [s] |
| arrow::write_csv_arrow | 23.3 [s] |
| arrow::write_parquet | 22.5 [s] |
| arrow::write_feather | 10.0 [s] |
| readr::write_csv | 6.6 [s] |
| data.table::fwrite | 6.2 [s] |

# Writing data - File sizes

| function | size |
|---|---|
| utils::write.csv | 1825.1 [megabytes] |
| arrow::write_csv_arrow | 1651.8 [megabytes] |
| readr::write_csv | 1432.5 [megabytes] |
| data.table::fwrite | 1378.0 [megabytes] |
| arrow::write_feather | 642.2 [megabytes] |
| arrow::write_parquet 50 partitions | 271.6 [megabytes] |
| base::save | 221.9 [megabytes] |
| arrow::write_parquet | 126.1 [megabytes] |

# Reading data

- utils::read.csv
- readr::read_csv
- data.table::fread
- arrow::read_csv_arrow
- arrow::read_feather
- arrow::read_parquet
- arrow::open_dataset multi-file parquet

# Reading data - utils::read.csv

```
tic()
df <- read.csv("../data/nla_big.csv")
toc()
```

```
38.12 sec elapsed
```

# Reading data - readr::read_csv

```
tic()
df <- read_csv("../data/nla_big.csv")
toc()
```

```
37.33 sec elapsed
```

# Reading data - data.table::fread

```
tic()
df <- fread("../data/nla_big.csv")
toc()
12.01 sec elapsed
```

# Reading data - arrow::read_csv_arrow

```
tic()
df <- read_csv_arrow("../data/nla_big.csv")
toc()
16.06 sec elapsed
```

# Reading data - arrow::read_feather

```
tic()
df <- read_feather("../data/nla_big.feather")
toc()
```

```
4.01 sec elapsed
```

# Reading data - arrow::read_parquet

```
tic()
df <- read_parquet("../data/nla_big.parquet")
toc()
12.39 sec elapsed
```

# Reading data - arrow::open_dataset multi-file parquet

```
tic()
df <- open_dataset("../data/nla_big")
toc()
```

```
0.17 sec elapsed
```

# Reading data - Times

| function | time |
|---|---|
| utils::read.csv | 38.1 [s] |
| readr::read_csv | 37.3 [s] |
| arrow::read_csv_arrow | 16.1 [s] |
| arrow::read_parquet | 12.4 [s] |
| data.table::fread | 12.0 [s] |
| arrow::read_feather | 4.0 [s] |
| arrow::open_dataset | 0.2 [s] |

# Summarizing data

- Take our 10 million rows and
  - group on state and analyte
  - provide state average for each analyte
  - count number of samples per group
  - Result:
    - 4 columns
      - state, analyte, avg_result, group_n
    - 919 rows

# Summarizing data

- readr::read_csv
- data.table::fread
- arrow::read_feather
- arrow::read_parquet
- arrow::open_dataset

# Summarizing data - readr::read_csv

```
tic()
df <- read_csv("../data/nla_big.csv")
df_f_sum <- df |>
  group_by(state, analyte) |>
  summarize(avg_result = round(mean(result), 2),
            group_n = n()) |>
  ungroup()
toc()
class(df)
```

```
48.48 sec elapsed
[1] "spec_tbl_df" "tbl_df"      "tbl"
"data.frame"
```

# Summarizing data - data.table::fread

```
tic()
df <- fread("../data/nla_big.csv")
df_f_sum <- df |>
  group_by(state, analyte) |>
  summarize(avg_result = round(mean(result), 2),
            group_n = n()) |>
  ungroup()
toc()
class(df)
```

```
13.31 sec elapsed
[1] "data.table" "data.frame"
```

# Summarizing data - arrow::read_feather

```
tic()
df <- read_feather("../data/nla_big.feather")
df_f_sum <- df |>
  group_by(state, analyte) |>
  summarize(avg_result = round(mean(result), 2),
            group_n = n()) |>
  ungroup()
toc()
class(df)
```

```
5.36 sec elapsed
[1] "tbl_df"     "tbl"        "data.frame"
```

# Summarizing data - arrow::read_parquet

```
tic()
df <- read_parquet("../data/nla_big.parquet")
df_f_sum <- df |>
  group_by(state, analyte) |>
  summarize(avg_result = round(mean(result), 2),
            group_n = n()) |>
  ungroup()
toc()
class(df)
```

```
11.17 sec elapsed
[1] "tbl_df"     "tbl"        "data.frame"
```

# Summarizing data - arrow::open_dataset parquet multiple partitions

```
tic()
df <- open_dataset("../data/nla_big")
df_f_sum <- df |>
  group_by(state, analyte) |>
  summarize(avg_result = round(mean(result), 2),
            group_n = n()) |>
  ungroup() |>
  collect()
toc()
class(df)
```

```
3.4 sec elapsed
[1] "FileSystemDataset" "Dataset"                "ArrowObject"
[4] "R6"
```

# Summarizing data - Times

| function | time |
|---|---|
| readr::read_csv | 48.5 [s] |
| data.table::fread | 13.3 [s] |
| arrow::open_dataset | 11.2 [s] |
| arrow::read_feather | 5.4 [s] |
| arrow::open_dataset multiple partitions | 3.4 [s] |

# Another example - NYC Taxi on S3

- OK, this might be “big” data…

```
library(arrow)
bucket <- s3_bucket("voltrondata-labs-datasets",
                    anonymous = TRUE,
                    region = 'us-east-2')
tic()
nyc_taxi <- open_dataset(bucket$path("nyc-taxi"))
nrow(nyc_taxi)
[1] 1672590319
toc()
159.16 sec elapsed
```

# Big Data Summarize - NYC Taxi on S3

```
tic()
years <- nyc_taxi |>
  group_by(year) |>
  summarize(n = n()) |>
  collect()
head(years, 2)
# A tibble: 2 × 2
   year         n
  <int>     <int>
1  2009 170896055
2  2010 169001153
toc()
51.05 sec elapsed
```

# Total time

- Jeff's EISD Laptop: 8 cores, 16GB RAM
  - `16.55 [min]`
- DMAP xLarge: 4 cores, 16GB RAM
  - `7.11 [min]`
- DMAP 4xLarge: 16 cores, 64GB RAM
  - `6.53 [min]`
- DMAP Mem Int - 16xLarge: 64 cores, 976GB RAM
  - `8.22 [min]`

# Jeff’s Hot Takes

- Default to feather
  - Smaller than csv
  - Fastest to read
  - Faster to write
- When use case requires switch to parquet
  - Smallest files
  - Partitions for smaller individual files
- DMAP is your friend (if you have access…)
  - Don’t always need tons of RAM