Using the Apache Arrow C++ Library in R

Efficient workflow with large, multi-file datasets

Devin Bunch
University of Oregon
2021-09-13

The `arrow` package

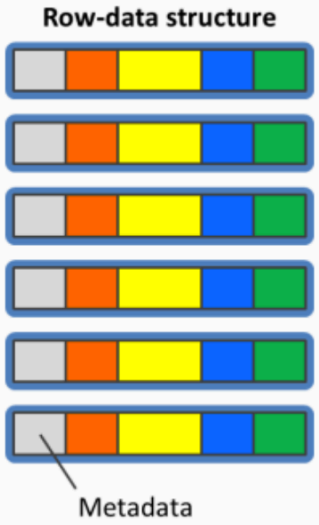
The `arrow` package

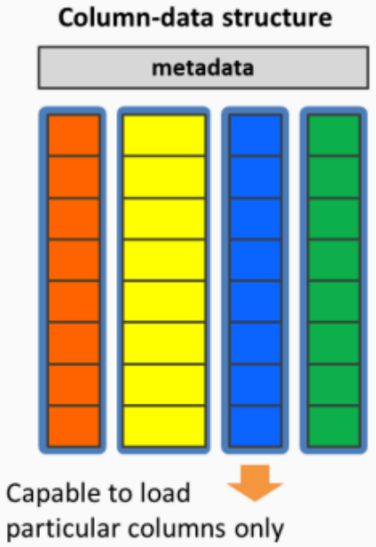
Objects

`arrow` returns two data structures with the same, **columnary format**:

1. **Table:** a tabular, column-oriented data structure capable of storing and processing large amounts of data more efficiently than R's built-in data.frame within databases

2. **Dataset:** a data structure with the capability to work on larger-than-memory data partitioned across multiple files



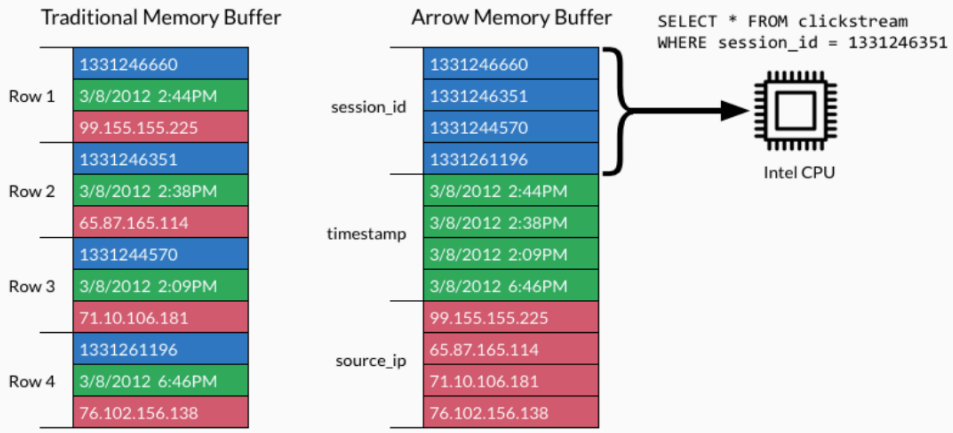


The `arrow` package

Benefits of Parquet files

1. Now we can **read parquet files** into R
2. Apache Parquet is **column-oriented**, meaning the values of each table column are stored next to each other, rather than all in the same row like `.csv` files
3. `arrow` provides major **R speedup** with functions for reading and writing large data files
4. We can apply the **same** familiar, user friendly `dplyr` verbs on **Arrow Table** objects

	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138



Columnar Structure

Using `dplyr` Syntax within `arrow`

Using `dplyr` Syntax within `arrow`

Reading and Writing Files

- The `arrow` package provides functions that return an R data.frame as defualt †
  - **`read_parquet()`** : read a Parquet file in columnary format
  - **`read_csv_arrow()`** : read a comma-separated values (CSV) file in row format
- **Functions** in `arrow` can be used with **R data.frame** and **Arrow Table** objects alike
  - Inside `dplyr` verbs, `arrow` offers support for common functions to get mapped to their base R and tidyverse equivalents if none exist

† To return an Arrow Table, set argument as_data_frame = FALSE

NYC Taxi Data Example

NYC Taxi Data Example

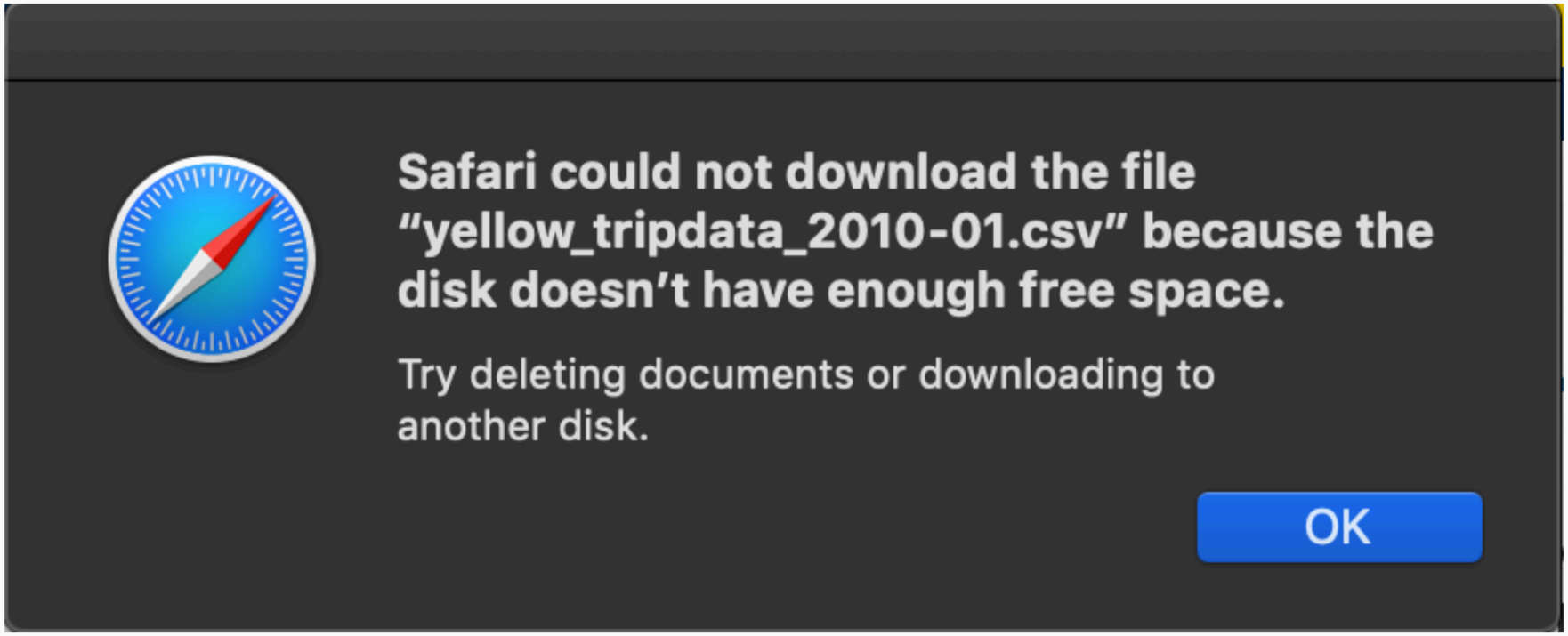
Why can't we just download the file locally?

NYC Taxi Data Example

Why can't we just download the file locally?

This metadata is **37** gigabytes in size . .

My storage error message:



Apache Arrow C++ Library Installation

Steps:

[_] **1. Install New Package** `arrow`

Apache Arrow C++ Library Installation

Steps:

[✅] 1. Install New Package `arrow`

```
## Load/install packages if necessary
if (!require("arrow")) install.packages("arrow")
library(arrow)
```

Apache Arrow C++ Library Installation

Steps:

[✅] 1. Install New Package `arrow`

```
## Load/install packages if necessary
if (!require("arrow")) install.packages("arrow")
library(arrow)
```

The `arrow` package takes care of all our **dependencies** needed for working with the **Apache Arrow C ++ Library**

. . . and that's all we have to do!

NYC taxi data

Metadata

We can use `arrow` to load in subsets of the full data

Requirements

To see if your arrow installation has S3 support, run

```
arrow::arrow_with_s3() ## Mine does!
```

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

Sync a local connection to the source of the parquet data file, externally

```
#arrow::copy_files("s3://ursa-labs-taxi-data", "nyc-taxi")
dir.exists("nyc-taxi") ## check to make sure it exists!
```

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

NYC taxi data

Load these explicit libraries in having acquired our knowledge,

```
library(arrow, warn.conflicts = FALSE)
library(dplyr, warn.conflicts = FALSE)
```

Now that we are synced, we can Create our Dataset object, pointing at the directory of data

```
ds ← open_dataset("nyc-taxi", partitioning = c("year", "month"))
```

NY taxi data

- Up to this point, we haven't loaded any data: we have walked directories to find files, we've parsed file paths to identify partition
- arrow supports the dplyr verbs mutate(), transmute(), select(), rename(), relocate(), filter(), and arrange().
- GOAL: pull the selected subset of the data into an in-memory R data frame

Let's find the median tip percentage for rides with fares greater than $100 in 2015, broken down by the number of passengers:

```
system.time(ds %>%
  filter(total_amount > 100, year == 2015) %>%
  select(tip_amount, total_amount, passenger_count) %>%
  mutate(tip_pct = 100 * tip_amount / total_amount) %>%
  group_by(passenger_count) %>%
  collect() %>%
  summarise(
    median_tip_pct = median(tip_pct),
    n = n()
  ) %>%
  print())
```

Results

```
##
## # A tibble: 10 x 3
##    passenger_count median_tip_pct      n
##              <int>          <dbl>  <int>
##  1               0           9.84    380
##  2               1          16.7  143087
##  3               2          16.6   34418
##  4               3          14.4    8922
##  5               4          11.4    4771
##  6               5          16.7    5806
##  7               6          16.7    3338
##  8               7          16.7      11
##  9               8          16.7      32
## 10               9          16.7      42
##
##    user  system elapsed
##   4.436   1.012   1.402
```

We just selected a subset out of a dataset with around **2 billion rows, computed a new column, and aggregated on it in under 2 seconds** on our laptops!

Applications are endless . . .

- Parquet is an open source file format available to any project in the **Hadoop ecosystem FOUND HERE**
- The Arrow Datasets library provides functionality to efficiently work with tabular, potentially larger than memory, and multi-file datasets.
- Link to learn more about the C ++ library languages: **https://www.tutorialspoint.com/c_standard_library/index.htm**
- Other applications of arrow are described in the following vignettes:
  vignette("python", package = "arrow"): use arrow and reticulate to pass data between R and Python
  vignette("flight", package = "arrow"): connect to Arrow Flight RPC servers to send and receive data
  vignette("arrow", package = "arrow"): access and manipulate Arrow objects through low-level bindings to the C++ library