Arrow for R : CHEAT SHEET

Arrow

Apache Arrow is a multi-language toolbox for accelerated data interchange and processing. It specifies a standardized language-independent column-based memory format for flat and hierarchical data, organized for efficient analytic operations on modern hardware.

The **arrow** R package provides access to the Arrow C++ library from R, and supplies an interface with **dplyr** and other familiar R functions.

Arrow Data Structures

Table: a tabular, column-oriented data structure capable of efficiently storing and processing large amounts of data with expanded column data types.

Dataset: similar to Table but with the capability to work on larger-than-memory data partitioned across multiple files.

You can convert an existing or object into an **arrow** , and an **arrow** to a or to view or work with the data in R:

data.frame tibble Table Table data.frame tibble

```
library(arrow)
library(dplyr)

df_table <- arrow_table(df)

df <- as.data.frame(df_table)
df <- as_tibble(df_table)
```

Read Individual Files

Read a data file from disk:

```
df <- read_parquet("file.parquet")
df <- read_feather("file.feather")
df <- read_csv_arrow("file.csv")
df <- read_json_arrow("file.json")
```

The **arrow** read_* functions return a data.frame , setting as_data_frame = FALSE returns an arrow Table

Read Multi-file Datasets

arrow defines Dataset objects for reading and writing very large files or sets of multi-files. The functions open_dataset() and write_dataset() enable analysis and processing of larger-than-memory data.

Read in multi-files from a directory:

```
open_dataset("folder")
```

Read in multi-files partitioned by year and month within a directory:

```
open_dataset("folder",
     partitioning = c("year", "month"))
```

The file format for open_dataset() is controlled by the format parameter, which has a default value of "parquet". Other supported formats include:

- "arrow"
- "feather" or "ipc" (aliases for "arrow")
- "csv" and "tsv"
- "text" (generic text-delimited files - use the delimiter argument to specify which to use)

Hive style partitioning is also supported, with partitions detected automatically from the file paths:

```
year=2021/month=12/data.parqu
et
year=2022/month=01/data.parqu
et
year=2022/month=02/data.parq
uet ...
```

Write Multi-file Datasets

Save partitioned data to disk based on columns in the data:

```
write_dataset(df, "data_partitioned",
  partitioning = c("year", "month"))
```

Default partitioning is based on any existing groups in the tibble or data.frame.

Write Individual Files

Write an R object df to disk:

```
write_parquet(df,     "file.parquet")
write_feather(df,      "file.feather")
write_csv_arrow(df, "file.csv")
```

To save a compressed file to disk, you specify the compression algorithm with the compression argument:

```
write_parquet(df, "file.parquet",
compression = "gzip")
```

Some file formats write compressed data by default. For more information on the supported compression algorithms see:

```
?write_parquet()
?write_feather()
?write_dataset()
```

Manipulate Larger-than-Memory Datasets

arrow lets you work efficiently with large, multi-file datasets by providing a **dplyr**—and many other R functions—interface to query, manipulate and summarise large datasets *before* pulling data into your R session with **dplyr** ' s : collect()

```
arrow_table(starwars) %>%
 filter(homeworld == "Tatooine") %>%
 rename(height_cm = height,
 mass_kg = mass) %>%
 mutate(height_in = height_cm / 2.54,
 mass_lbs = mass_kg * 2.2046) %>%
 arrange(desc(birth_year)) %>%
 select(name, height_in, mass_lbs) %>%
 collect()
```

In addition to most single-table **dplyr** verbs, many other function mappings are implemented in **arrow**, including base R, **lubridate**, and **stringr** functions.

Arrow for R : CHEAT SHEET

Manipulate Larger-than-Memory Datasets (cont)

arrow supports joins for joining multiple tables:

```
robot <- data.frame(
  species = c("Human", "Droid", "Ewok"),
bot = c(FALSE, TRUE, FALSE)
)

arrow_table(starwars) %>%
 select(name, species) %>%
 left_join(robot) %>%
 collect()
```

If you use **arrow** with partitioned data, **arrow** will only read from the relevant partitions:

```
year=2021/month=12/data.parquet
year=2022/month=01/data.parquet
year=2022/month=02/data.parquet
...
```

```
open_dataset("folder",
 partitioning = c("year", "month")) %>%
filter(year == 2022) %>%
group_by(month) %>%
summarise(total = sum(amount)) %>%
collect()
```

Table

For queries on objects, if **arrow** detects an unimplemented function, it will automatically call `collect()` and pull the data into R with a warning message:

```
## Warning: Expression
unimplemented_function() not supported in
Arrow;
## pulling data into R
```

Dataset

For queries on objects (which can be larger than memory), if **arrow** detects an unimplemented function, it will raise an error. You will need to explicitly tell **arrow** to `collect()` *before* the unimplemented function.

Zero-Copy R and Python Data Sharing

arrow provides **reticulate** methods for passing data between R and Python using the Python **pyarrow** library. Install, load, and import **pyarrow** in a virtual environment:

```
library(reticulate)

virtualenv_create("arrow-env")
install_pyarrow("arrow-env")
use_virtualenv("arrow-env")
pa <- import("pyarrow")
```

Use **pyarrow** to create an Arrow array object in an R session:

```
a <- pa$array(c(1, 2, 3))
```

Call a **pyarrow** function from your R session:

```
table1 <- arrow_table(starwars[1:5,])
table2 <- arrow_table(starwars[11:15,])

pa$concat_tables(tables = list(table1,
table2)) %>%
 collect()
```

Transport Data with Flight

Connect to Flight RPC server to send and receive data with **arrow**:

```
library(reticulate)
library(arrow)
install_pyarrow()

demo <- load_flight_server("flight_server")
server <- demo$FlightServer(port = 8089)
server$serve()

client <- flight_connect(port = 8089)

flight_put(client, df, path = "data/df")
df <- flight_get(client, "data/df")
```

Cloud Storage Support (S3)

Arrow supports reading files and multi-file datasets from cloud storage without having to download them first—`open_dataset()`, `write_dataset()` and **arrow**'s `read_*` and `write_*` functions all accept an S3 Uniform Resource Identifier (URI) as the source or destination file.

Read a file, a multi-file dataset, or partitioned multi-file dataset:

```
df <- read_parquet(
 "s3://ursa-labs-taxi-data/2019/
 06/data.parquet")

df <-
open_dataset("s3://ursa-labs-taxi-data")

df <- open_dataset(
 "s3://ursa-labs-taxi-data",
partitioning = c("year", "month"))
```

Create an **arrow** `FileSystem` object and pass that to **arrow**'s read and write functions:

```
bucket <- s3_bucket("ursa-labs-taxi-data")
df <- read_parquet(bucket$path(
 "2019/06/data.parquet"))
```

Copy data from cloud storage to your computer:

```
copy_files("s3://ursa-labs-taxi-data",
"~/nyc-taxi")
```

Detailed instructions for working with S3 cloud storage are available here: arrow.apache.org/docs/r/articles/fs

Additional Resources

Arrow R Cookbook: arrow.apache.org/cookbook/r/

Reference guide to **arrow** in R: arrow.apache.org/docs/r/

Arrow communities: arrow.apache.org/community/