Big data in R with arrow :: CHEAT SHEET

About

Apache Arrow is a development platform for in-memory analytics. It contains a set of technologies that enable big data systems to process and move data fast.

The arrow R package integrates with dplyr and allows you to work with multiple storage formats as well as data in AWS S3 and other similar cloud storage systems.

Installation

To install from CRAN,

```
install.packages("arrow")
```

To get all optional features enabled on Linux, you should set the environment variable `NOT_CRAN=true` prior to installing.

MacOS and Windows binary packages include all of these features by default.

Conda users can install with

```
conda install -c conda-forge --
strict-channel-priority r-arrow
```

Visit **apache.arrow.org** for the specifics to install the development version.

Import

For **single files** you can do either:

```
read_parquet("gapminder.parquet")
read_feather("gapminder.feather")
```

Arrow can also read large CSV and JSON files with excellent speed and efficiency:

```
read_csv_arrow("gapminder.csv")
read_json_arrow("gapminder.json")
```

This reads data as data.frame.

To read **multiple Parquet/Feather files** from a directory you can specify a partitioning for filtering:

```
d <- open_dataset("nyc-taxi",
partitioning = c("year",
"month"))
```

This reads data as ArrowObject and needs posterior steps.

Dplyr compatibility

Arrow in R shares most of the characteristics of SQL in R throught RPostgres and other packages. The use of `collect()` converts Arrow-type objects into regular data.frames and allows you to use your data with your existing visualisation and analysis workflow.

If an operation is not yet implemented in Arrow, you can `collect()` and then do the operation on the R data.frame. This selection of data does need to fit into memory, but the whole dataset does not, you just need to use Arrow to select/filter down to something that does.

dplyr shall read exactly the required data fragments after you specified a partitioning without the need for an index as in SQL, here's an example of this:

```
d2 <- d %>%
 filter(year == 2009,
  month == 1) %>%
 collect() %>%
 group_by(year,month) %>%
 summarise(mean_amount =
 mean(total_amount))
```

This takes an ArrowObject and creates a data.frame.

Export

To save without partitioning, you can use:

```
write_parquet(d2, "d2.parquet")
write_feather(d2, "d2.feather")
write_csv_arrow(d2, "d2.csv")
```

This is very similar to `write_csv()` from readr.