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.

Big data in R with arrow :: CHEAT SHEET

Export (continued)

When saving data stored in a tibble to parquet format, the default partitioning is based on any groups in the tibble.

To save with partitioning:

```
d2 %>%
 group_by(year, month) %>%
 write_dataset("nyc-summary",
 hive_style = TRUE)
```

This creates the folder 2009/01 with the saved data. Experiment changing hive to FALSE or changing the output to feather or csv.

Amazon S3 support

You can read files from S3 file systems without having to download them first.

One starting point is to list the folders in the S3 space:

```
taxi_s3 <- s3_bucket("s3://ursa-
labs-taxi-data")
taxi_s3$ls()
```

This shows the folders and files at the top directory. From here you can list a certain folder contents like `taxi_s3$ls("2009", recursive = TRUE)`.

After listing the files you can open a single file exactly as for local files:

```
d <- read_parquet("s3://ursa-
labs-taxi-data/2009/01/
data.parquet")
```

Another option is to read from a directory:

```
d2 <- open_dataset("s3://ursa-
labs-taxi-data", partitioning =
c("year", "month"))
```

However, reading from S3 is not optimal unless you're in EC2, otherwise the network speed will make this very slow.

You can also copy the data to your computer:

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

Remember that you can always use tools such as rclone to copy/sync data back and forth.

Generic S3 support

The same methods as for Amazon S3 apply, but opening a connection requires additional commands.

You can read or copy from different S3 compatible providers such as DigitalOcean and many others.

In order to connect to a generic S3 you can adapt this example:

```
taxi_s3  <- S3FileSystem$create(
  access_key =
  Sys.getenv('ACCESS_KEY'),
  secret_key =
  Sys.getenv('SECRET_KEY'),
  scheme = "https",
  endpoint_override =
  "sfo3.digitaloceanspaces.com"
)
```

Additional resources

Don't forget to read the documentation on **arrow.apache.org**.

You can ask on Stackoverflow under the r/apache-arrow tags or on GitHub (github.com/apache/arrow).

We have a very active community with daily email communication. Please send an email to **user-subscribe@arrow.apache.org** with the subject "Subscribe" to stay connected. Email us your questions with a subject such as "[R] problem with group by".