Working with tabular data

Page maintainers: Suvayu Ali @suvayu , Flavio Hafner @f-hafner and Reggie Cushing @recap

There are several solutions available to you as an RSE, with their own pros and cons. You should evaluate which one works best for your project, and project partners, and pick one. Sometimes it might be, that you need to combine two different types of technologies. Here are some examples from our experience.

You will encounter datasets in various file formats like: - CSV/Excel - Parquet - HDF5/NetCDF - JSON/JSON-LD

Or local database files like SQLite. It is important to note, the various trade-offs between these formats. For instance, doing a random seek is difficult with a large dataset for non-binary formats like: CSV, Excel, or JSON. In such cases you should consider formats like Parquet, or HDF5/NetCDF. Non-binary files can also be imported into local databases like SQLite or DuckDB. Below we compare some options to work with datasets in these formats.

It's also good to know about Apache Arrow, which is not itself a file format, but a specification for a memory layout of (binary) data. There is an ecosystem of libraries for all major languages to handle data in this format. It is used as the back-end of many data handling projects, among which a few others mentioned in this chapter.

Local database

When you have a relational dataset, it is recommended that you use a database. Using local databases like SQLite and DuckDB can be very easy because of no setup requirements. But they come with some some limitations; for instance, multiple users cannot write to the database simultaneously.

SQLite is a transactional database, so if you have a dataset that is changing with time (e.g. you are adding new rows), it would be more appropriate. However in research often we work with static databases, and are interested mostly in analytical tasks. For such a case, DuckDB is a more appropriate alternative. Between the two, - DuckDB can also create views (virtual tables) from other sources like files, other databases, but with SQLite you always have to import the data before running any queries. - DuckDB is multi-threaded. This can be an advantage for large databases, where aggregation queries tend to be faster than sqlite. - However if you have a really large dataset, say 100Ms of rows, and want to perform a deeply nested query, it would require substantial amount of memory, making it unfeasible to run on personal laptops. - There are options to customize memory handling, and push what is possible on a single machine.