Working with external databases and SQL

Introduction to Data Visualization

Table of Contents

1 README

This lecture gives a brief introduction relational databases, with an emphasis on either interacting with them *from* R, or using them *within* R via the `sqldf` and `RSQLite` packages. Largely based on: Pearson, chapter 4 (2016).

For a practice file, download tinyurl.com/2p8e6se5

2 Why databases and not spreadsheets

- A "flat file" representation (like CSV) often has many records with the same values for certain fields, i.e. high search overhead
- Relational databases can store very large data volumes efficiently ("normalized") as collection of linked tables
- Because of this organisation to update a customer's name in a database, you only need to update information (e.g. address) once
- Relational database management systems (RDBMS) allow fast, concurrent access to a large number of records stored in linked tables and multiple databases

3 SQL - Structured Query Language

- SQL (Structured Query Language) is the language to define, manipulate, query and control RDBMS

- SQL queries are "natural", e.g. to select a table column 'address' of a customer named 'Smith' from a table 'customer', write:

```
SELECT address
  FROM customer
  WHERE name = 'Smith'
```

- Here is the 'hello world' program in SQLite. The code block opens on an SQLite database `rsqlite.db`, which is created if it does not exist.

```
SELECT 'hello world'
```

- SQL has great query abilities, including statistical functions, but no visualization capability whatsoever. A little SQL plus R guarantees your data science success

4 Database creation and use

1. Databases are designed like data frames - variables of specific types are stored in rectangular tables

2. The database design is implemented in a specific software environment - examples are Oracle, MySQL, PostgreSQL, Microsoft SQL Server, or SQLite.

   Among these, SQLite is the most common database of all: it is a free, fast, lightweight RDBMS written in C that can live on "bare metal" computing devices (cars, sensors, microcontrollers etc.). It is present in every mobile phone and mobile device.

3. Users run SQL queries against the database to extract the specific subsets of data they need and/or basic statistical functions.

5 Interaction with databases using R

R users are typically consumers of data from databases that have been created by others. Querying with R includes:

1. Connect to the database from your R session;
2. Execute SQL queries against the database to retrieve the data we want, frequently involves:
   - a merge of records distributed over several tables (so-called `INNER JOIN`)
   - functions like `AVG` (average), `SUM` or `COUNT` to sum or count number of records, `MIN` and `MAX` to find minimal and maximal values
   - a grouping of records with `GROUP BY` and run subqueries on the groups
   - combine important or frequently used sets of commands as procedures (SQL scripts)
3. Close the database connection.

Here, steps 1,3 depend on the database implementation while 2 is fairly standardized because of the maturity and age of SQL.

6 The `sqldf` package

- The `sqldf` package supports SQL queries directly against data frames.
- It also includes the function `read.csv.sql`, with which you can run SQL queries directly against external CSV files. Useful when you want to get information from a small part of an enormous CSV file.

- We use the `auto-mpg` dataset that can be obtained from the UCI Machine Learning Repository including unusual cars with 3 and 5 cylinders.
- You can get this dataframe by running the code in the `9_internet.org` file, available from GitHub here: tinyurl.com/cfwhayya, with `M-x org-babel-execute-buffer`

- Or you can read this CSV file into a dataframe `autoMpgFrame` - the CSV file was created with the command `write.csv(autoMpgFrame, file="autoMpgFrame.csv")`: tinyurl.com/2tx2sr2z

```
autoMpgFrame <- read.csv(file="https://tinyurl.com/2tx2sr2z",     header=TRUE)
autoMpgFrame <- autoMpgFrame[,-1]
str(df)
```

```
function (x, df1, df2, ncp, log = FALSE)
```

- You should now have the dataframe `autoMpgFrame` loaded:

```
ls(autoMpgFrame) # lists the dataframe variables
```

```
[1] "acceleration" "carName"      "cylinders"    "displacement" "horsepower"
[6] "modelYear"    "mpg"          "origin"       "weight"
```

- Install the `sqldf` package from the R console `*R*`:

  1. when you are asked if you `want to install from sources the package which needs compilation`, say `no`.
  2. the package dependencies include `RSQLite` and `DBI`, two packages that we will use below.

```
install.packages("sqldf")  # you only need to do this once
```

7 Filtering records with `WHERE`

- The following query uses SQL to extract the number of `cylinders`, the `modelYear` and the `carName` for all of these unusual cars:

  1. The first line loads the package (and its 3 dependencies)
  2. The `sqldf` function is called with an SQL query
  3. The SQL query selects the required variables from the dataframe with the desired condition.

```
library(sqldf)  # loads required 'dependencies' (linked libraries)

strangeCars <- sqldf(
  "SELECT cylinders, modelYear, carName
        FROM autoMpgFrame
        WHERE cylinders == 3 OR cylinders == 5")

strangeCars
```

```
Loading required package: gsubfn
Loading required package: proto
Loading required package: RSQLite
  cylinders modelYear             carName
1         3        72     mazda rx2 coupe
2         3        73           maxda rx3
3         3        77          mazda rx-4
4         5        78           audi 5000
5         5        79  mercedes benz 300d
6         5        80 audi 5000s (diesel)
7         3        80       mazda rx-7 gs
```

- The result is a regular dataframe:

```
str(strangeCars)
```

```
'data.frame':   7 obs. of  3 variables:
 $ cylinders: int  3 3 3 5 5 5 3
 $ modelYear: int  72 73 77 78 79 80 80
 $ carName  : chr  "mazda rx2 coupe" "maxda rx3" "mazda rx-4" "audi 5000" ...
```

8 Grouping records with `GROUP BY`

- In the next example, we use `GROUP BY` to retrieve as function of the number of `cylinders`:

  1. the average (`AVG`) mileage (`mpg`),
  2. `horsepower`,
  3. `weight`,
  4. number of records (`COUNT`) in each group

```
cylinderSummary <- sqldf(
  "SELECT cylinders as cyl,
          AVG(mpg) as mean_mpg,
          AVG(horsepower) as mean_hp,
          AVG(weight) as mean_weight,
          COUNT(*) as cars
     FROM autoMpgFrame
     GROUP BY cylinders")
cylinderSummary
```

```
  cyl mean_mpg   mean_hp mean_weight cars
1   3 20.55000  99.25000    2398.500    4
2   4 29.28676  78.28141    2308.127  204
3   5 27.36667  82.33333    3103.333    3
4   6 19.98571 101.50602    3198.226   84
5   8 14.96311 158.30097    4114.718  103
```

- The result is a regular dataframe whose `names` are aliases given to the return vectors inside the `SELECT` command:

```
str(cylinderSummary); names(cylinderSummary)
```

```
'data.frame':   5 obs. of  5 variables:
 $ cyl        : int  3 4 5 6 8
 $ mean_mpg   : num  20.6 29.3 27.4 20 15
 $ mean_hp    : num  99.2 78.3 82.3 101.5 158.3
 $ mean_weight: num  2398 2308 3103 3198 4115
 $ cars       : int  4 204 3 84 103
[1] "cyl"         "mean_mpg"    "mean_hp"     "mean_weight" "cars"
```

- The wildcard argument for `COUNT` works because the table is rectangular: `COUNT(cylinders)` would also have worked, since every column has the same length of records.

9 TODO `INNER JOIN` between dataframes

- This content is reserved for the advanced introduction to data science course because queries with merged tables require a little more SQL than I can convey in 50 minutes

10 Database support in R with `DBI` and `RODBC`

- Databases represent external files usually hosted on external servers (other computers), accessed over a network
- To work with a database from an interactive R session, you must:
  1. Connect to the database
  2. Execute SQL commands on the database - these can include creating or deleting tables (e.g. `CREATE TABLE`) and manipulating table content (e.g. `INSERT INTO`) or structure (e.g. `ALTER TABLE`)
  3. Disconnect from the database
- Database (DB) communication support is provided by two packages: `DBI` or `RODBC` (for the Open Database Connectivity standard - RDBMS like MySQL, Microsoft SQL Server, Microsoft Access, etc.)
- `DBI` supports
  - Oracle through the `ROracle` package,
  - PostgreSQL through the `RPostgreSQL` package, and
  - SQLite through the `RSQLite` package
- The key functions provided by `DBI` are:
  1. `dbDriver` to specify the DB type, e.g. `dbDriver("PostgreSQL")`
  2. `dbConnect` to connect with a specific DB
  3. `dbGetQuery` to send SQL queries to the DB and retrieve results
  4. `dbDisconnect` terminates our connection with the DB

11 The `RSQLite` package

- In our `RSQLite` example, we
  1. create a new `sqlite3` database
  2. connect to the database
  3. create new tables in the database
  4. check what the database contains
  5. apply SQL queries against the database
  6. disconnect from the database
- `RSQLite` was already installed if you installed `sqldf`, otherwise you must install it before you can use it.

12 Create and connect to an SQLite database

- After loading the package, we create an SQLite database `mtcars.db`. If it does not exist it will be created.

```
library(RSQLite)
conn <- dbConnect(SQLite(), "../data/mtcars.db")
```

- We can check if this database was created using `shell` (your relative path to the file may be different) - and if it is empty:

```
shell('DIR/W "../data/mtcars.db"')
```

```
 Volume in drive C is OS
 Volume Serial Number is 0654-135C

 Directory of C:\Users\data

File Not Found
Warning message:
In shell("DIR/W \"../data/mtcars.db\"") :
  'DIR/W "../data/mtcars.db"' execution failed with error code 1
```

- What is the nature of this object `conn` that represents a connection?

```
class(conn)
```

```
Error: object 'conn' not found
```

```
str(conn)
```

```
Error in str(conn) : object 'conn' not found
```

- We learn that `conn` is a connection, and that it knows about `mtcars.db` and its absolute path address. This path could also be a network path.

13 Create a table in the database

- We need to put something in the new database - we use the existing, pre-loaded, well-known data frame `mtcars`.

```
dbWriteTable(conn, "MTCARSTABLE", mtcars)
```

- SQLite is not case-sensitive, I only write the commands in upper case to distinguish them from R (which is case-sensitive)

- Let's see if `mtcars.db` has changed (with a time option):

```
shell('DIR/T "../data/mtcars.db"')
```

```
 Volume in drive C is OS
 Volume Serial Number is 0654-135C

 Directory of C:\Users\data

File Not Found
Warning message:
In shell("DIR/T \"../data/mtcars.db\"") :
  'DIR/T "../data/mtcars.db"' execution failed with error code 1
```

14 Check SQLite database content

- We want to list the table in the database (accessed via `conn`):

```
dbListTables(conn)
```

```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbListTables
```

- We also want to know which fields this table contains:

```
dbListFields(conn, "MTCARSTABLE")
```

```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbListFields
```

- Everything seems to be there. But to know for sure, we need to `SELECT` the fields.

15 Query SQLite database

- To see how many records are in the table, we can use the `SELECT COUNT(*)` SQL statement on the table - displayed with an alias name:

```
dbGetQuery(conn,
           "SELECT COUNT(*) AS 'mtcars_count'
            FROM MTCARSTABLE")
```

```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbGetQuery':
```

- The `SELECT` command is very versatile - it can even do arithmetic (without using any tables - an empty database would suffice):

```
dbGetQuery(conn,
           "SELECT 1+1")
```

```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbGetQuery':
```

- To do something more interesting, let's repeat the `sqldf` example for `mtcars`

```
dbGetQuery(conn,
           "SELECT cyl,
                   AVG(mpg) as 'mean_mpg',
                   AVG(hp) as 'mean_hp',
                   AVG(wt) as 'mean_weight',
                   COUNT(*) as 'cars'
            FROM MTCARSTABLE
            GROUP BY cyl")
```

```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbGetQuery':
```

16 Close SQLite database connection

- Check your workspace before and after disconnecting from the database:

```
ls()
dbDisconnect(conn)
ls()
```

```
 [1] "autoMpgFrame"          "autoMpgNames"          "autoMpgRecords"
 [4] "ConvertAutoMpgRecords" "cylinderSummary"       "LatteIndexFrame"
 [7] "strangeCars"           "URL"                   "URL1"
[10] "URL2"                  "x"
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbDisconnect
 [1] "autoMpgFrame"          "autoMpgNames"          "autoMpgRecords"
 [4] "ConvertAutoMpgRecords" "cylinderSummary"       "LatteIndexFrame"
 [7] "strangeCars"           "URL"                   "URL1"
[10] "URL2"                  "x"
```

- The database connection is a so-called `S4` object - it is not part of your environment (but instead connects it to the external world). Consequently, `conn` is still there but it's not active:

```
dbListTables(conn)
```

```
Error in h(simpleError(msg, call)) :
  error in evaluating the argument 'conn' in selecting a method for function 'dbListTables
```

17 Further study

1. SQLite in R - lightweight DataCamp tutorial
    - Creating databases and tables
    - Executing SQL queries through RSQLite
    - Insert variables into queries
    - Administrating databases from R
2. A Comprehensive Introduction to Working with Databases using R
    - Free tutorial looking at connecting to databases from R
    - Uses the table manipulation package `dplyr` ("Tidyverse")
    - Overuse of "Tidyverse" packages overcomplicates thingsx

18 References

- Pearson RK (2016). Exploratory Data Analysis. CRC Press.

Author: Marcus Birkenkrahe
Created: 2022-11-20 Sun 16:16