R with SQLite RDBMS

Dr. Zahid Ansari

Table of Contents

```
install.packages("rmarkdown")
install.packages("tinytex")
library(tinytex)
install_tinytex()
```

1 SQLite with R

In this section, we will query a SQLite database installed on a local, personal computer directly from RStudio.

1.1 Preparing and importing data into a local SQLite database

SQLite is, by default, included in some distributions of popular operating systems, for example Mac OS X and in Windows 10. You can easily check whether your machine has SQLite installed by starting it through a Terminal/shell window:

```
C:\Users\ASUS>sqlite3
```

SQLite version 3.39.4 2022-09-29 15:55:41 Enter ".help" for usage hints. Connected to a transient in-memory database. Use ".open FILENAME" to reopen on a persistent database. sqlite>

- If the command produces the preceding output (or similar) your machine is already equipped with SQLite database.
- If for some reason your operating system does not contain SQLite, visit http://www.sqlite.org/download.html to download and install the binaries for your particular OS.
- Once installed, in a Terminal/shell navigate to the directory with our data, need_puf_2014.csv for example:

```
$ cd ~/Desktop/B05396_Ch06_Code/
```

- The data that we are going to use in this tutorial is the National Energy Efficiency Data – Framework: anonymised data 2014 (NEED) dataset provided by the Department of Energy & Climate Change.

- The NEED data includes household-level information on annual consumption of gas and electricity across different geographical locations in the United Kingdom, and covers the years from 2005 until 2012.

- The supplied Public Use File (PUF) is available for download from the following website: https://www.gov.uk/government/statistics/national-energyefficiency-data-framework-need-anonymised-data-2014.

- The NEED PUF file comes in a comma-separated (CSV) format (7.48 MB) and it contains a small representative sample of 49,815 records drawn from a full Big Data dataset of 4,086,448 records (1.38 GB as a SPSS *.sav file or 719 MB as a tab-delimited file) available to download through End-User License from the already introduced UK Data Archive at: https://discover.ukdataservice.ac.uk/catalogue/?sn=7518.

- Once in the directory with the data file, start SQLite by creating a new database called need_data:

```
D:\AMU Computer Science\Courses\Big Data Analytics\Big Data Analytics Using
R\Ch5>sqlite3 need_data
SQLite version 3.39.4 2022-09-29 15:55:41
Enter ".help" for usage hints.
sqlite>
```

- Type .databases to show all currently available databases:

```
sqlite> .databases
main: D:\AMU Computer Science\Courses\Big Data Analytics\Big Data Analytics
Using R\Ch5\need_data r/w
sqlite>
```

- At this stage, you may want to physically open the folder where your data is stored. You may now notice that a new empty file called need_data has been created.

- Then set the column separator to comma and import the need_puf_2014.csv data file as a new table called need:

```
sqlite> .separator ","
sqlite> .import need_puf_2014.csv need

D:\AMU Computer Science\Courses\Big Data Analytics\Big Data Analytics Using
R\Ch5>sqlite3 need_data
SQLite version 3.39.4 2022-09-29 15:55:41
Enter ".help" for usage hints.
sqlite> .separator ","
sqlite> .import need_puf_2014.csv need
sqlite>
```

- You can check the available tables by running the .tables command:

```
sqlite> .tables
need
sqlite>
```

- We can now view the folder with our data again. The need_data file has now been populated with the imported data.

- Also, we may inspect the structure of the table using a PRAGMA statement:

```
sqlite> PRAGMA table_info('need');
```

0,HH_ID,TEXT,0,,0 1,REGION,TEXT,0,,0 2,IMD_ENG,TEXT,0,,0 3,IMD_WALES,TEXT,0,,0 4,Gcons2005,TEXT,0,,0 5,Gcons2005Valid,TEXT,0,,0 …#output truncated

- The .schema function allows us to print the schema of the table. A schema is a structure of a database object, for example the names of variables, their classes, and other features of a table. In other words, the schema describes the design of the table.

- The following snippet creates a schema for our need table:

```
sqlite> .schema need
CREATE TABLE IF NOT EXISTS "need"(
"HH_ID" TEXT, "REGION" TEXT, "IMD_ENG" TEXT, "IMD_WALES" TEXT,
 "Gcons2005" TEXT, "Gcons2005Valid" TEXT, "Gcons2006" TEXT, "Gcons2006Valid"
TEXT,
 "Gcons2007" TEXT, "Gcons2007Valid" TEXT, "Gcons2008" TEXT, "Gcons2008Valid"
TEXT,
 "Gcons2009" TEXT, "Gcons2009Valid" TEXT, "Gcons2010" TEXT, "Gcons2010Valid"
TEXT,
 "Gcons2011" TEXT, "Gcons2011Valid" TEXT, "Gcons2012" TEXT, "Gcons2012Valid"
TEXT,
 "Econs2005" TEXT, "Econs2005Valid" TEXT, "Econs2006" TEXT, "Econs2006Valid"
TEXT,
 "Econs2007" TEXT, "Econs2007Valid" TEXT, "Econs2008" TEXT, "Econs2008Valid"
TEXT,
 "Econs2009" TEXT, "Econs2009Valid" TEXT, "Econs2010" TEXT, "Econs2010Valid"
TEXT,
 "Econs2011" TEXT, "Econs2011Valid" TEXT, "Econs2012" TEXT, "Econs2012Valid"
TEXT,
 "E7Flag2012" TEXT, "MAIN_HEAT_FUEL" TEXT, "PROP_AGE" TEXT, "PROP_TYPE" TEXT,
 "FLOOR_AREA_BAND" TEXT, "EE_BAND" TEXT, "LOFT_DEPTH" TEXT, "WALL_CONS" TEXT,
 "CWI" TEXT, "CWI_YEAR" TEXT, "LI" TEXT, "LI_YEAR" TEXT,
 "BOILER" TEXT, "BOILER_YEAR" TEXT);
sqlite>
```

- Once the table is created and our data is in it, we can open the RStudio application and connect to the SQLite database

1.2 Connecting to SQLite from RStudio

- When in RStudio, make sure that your working directory is set to the directory with the need_data file.

```
setwd("D:/AMU Computer Science/Courses/Big Data Analytics/Big Data Analytics 
Using R/Ch5")
```

- As R requires the RSQLite package to make a connection with the SQLite database using the DBI package, we have to download the new versions of DBI and its dependency Rcpp beforehand.
- Note that in order to install recent releases of these packages from GitHub repositories, you first need to install the devtools package – it allows connectivity with GitHub:

```
# install.packages("devtools")
library(devtools)
```

- Then, install and load the RSQLite package:

```
install.packages("RSQLite")

library(RSQLite)
```

- Let's then create a connection with the need_data SQLite database:

```
con <- dbConnect(SQLite(), "need_data")
#con <- dbConnect(RSQLite::SQLite(), "need_data")
con

## <SQLiteConnection>
##   Path: D:\AMU Computer Science\Courses\Big Data Analytics\Big Data 
Analytics Using R\Ch5\need_data
##   Extensions: TRUE
```

- The dbListTables() functions provides information on the available tables in the connected database

```
dbListTables(con)

## [1] "myneed"         "need"             "query_2_result"
```

- The dbListFields() function provides information on the available columns within a specified table

```
dbListFields(con, "need")

##  [1] "HH_ID"           "REGION"           "IMD_ENG"          "IMD_WALES"
##  [5] "Gcons2005"       "Gcons2005Valid"   "Gcons2006"        
"Gcons2006Valid"
##  [9] "Gcons2007"       "Gcons2007Valid"   "Gcons2008"        
"Gcons2008Valid"
## [13] "Gcons2009"       "Gcons2009Valid"   "Gcons2010"        
"Gcons2010Valid"
## [17] "Gcons2011"       "Gcons2011Valid"   "Gcons2012"        
"Gcons2012Valid"
```

```
## [21] "Econs2005"       "Econs2005Valid"  "Econs2006"
"Econs2006Valid"
## [25] "Econs2007"       "Econs2007Valid"  "Econs2008"
"Econs2008Valid"
## [29] "Econs2009"       "Econs2009Valid"  "Econs2010"
"Econs2010Valid"
## [33] "Econs2011"       "Econs2011Valid"  "Econs2012"
"Econs2012Valid"
## [37] "E7Flag2012"      "MAIN_HEAT_FUEL"  "PROP_AGE"        "PROP_TYPE"
## [41] "FLOOR_AREA_BAND" "EE_BAND"         "LOFT_DEPTH"      "WALL_CONS"
## [45] "CWI"             "CWI_YEAR"        "LI"              "LI_YEAR"
## [49] "BOILER"          "BOILER_YEAR"
```

- We may now query the data using the dbSendQuery() function;
- for example, we can retrieve all records from the table for which the value of the FLOOR_AREA_BAND variable equals 1:

```
query.1 <- dbSendQuery(con, "SELECT * FROM need WHERE FLOOR_AREA_BAND = 1")
dbGetStatement(query.1)

## [1] "SELECT * FROM need WHERE FLOOR_AREA_BAND = 1"
```

- In case you want to extract the string representing the SQL query used, you may apply the dbGetStatement() function on the object created by the dbSendQuery() command, as shown in the preceding code.

- The results set may now be easily pulled to R (note that all queries and data processing activities run directly on the database, thus saving valuable resources normally used by R processes): 589973

```
query.1.res <- fetch(query.1, n=50)
str(query.1.res)

## 'data.frame':    50 obs. of  50 variables:
##  $ HH_ID          : chr  "5" "6" "12" "27" ...
##  $ REGION         : chr  "E12000003" "E12000007" "E12000007" "E12000004"
...
##  $ IMD_ENG        : chr  "1" "2" "1" "1" ...
##  $ IMD_WALES      : chr  "" "" "" "" ...
##  $ Gcons2005      : chr  "" "" "" "5500" ...
##  $ Gcons2005Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2006      : chr  "" "" "" "4500" ...
##  $ Gcons2006Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2007      : chr  "" "" "" "8000" ...
##  $ Gcons2007Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2008      : chr  "" "" "" "5000" ...
##  $ Gcons2008Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2009      : chr  "" "" "" "4500" ...
##  $ Gcons2009Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2010      : chr  "" "" "" "5000" ...
##  $ Gcons2010Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2011      : chr  "" "" "" "12600" ...
##  $ Gcons2011Valid : chr  "M" "O" "M" "V" ...
##  $ Gcons2012      : chr  "" "" "" "7500" ...
##  $ Gcons2012Valid : chr  "M" "O" "M" "V" ...
##  $ Econs2005      : chr  "2500" "5000" "3000" "700" ...
##  $ Econs2005Valid : chr  "V" "V" "V" "V" ...
##  $ Econs2006      : chr  "2600" "4850" "2400" "" ...
##  $ Econs2006Valid : chr  "V" "V" "V" "L" ...
##  $ Econs2007      : chr  "" "4900" "1900" "1800" ...
##  $ Econs2007Valid : chr  "L" "V" "V" "V" ...
##  $ Econs2008      : chr  "1200" "4100" "1550" "950" ...
##  $ Econs2008Valid : chr  "V" "V" "V" "V" ...
##  $ Econs2009      : chr  "2300" "4300" "1600" "600" ...
##  $ Econs2009Valid : chr  "V" "V" "V" "V" ...
##  $ Econs2010      : chr  "2650" "800" "1450" "400" ...
##  $ Econs2010Valid : chr  "V" "V" "V" "V" ...
##  $ Econs2011      : chr  "2800" "2000" "2150" "2700" ...
##  $ Econs2011Valid : chr  "V" "V" "V" "V" ...
##  $ Econs2012      : chr  "1950" "3900" "1200" "1100" ...
##  $ Econs2012Valid : chr  "V" "V" "V" "V" ...
##  $ E7Flag2012     : chr  "" "1" "" "" ...
##  $ MAIN_HEAT_FUEL : chr  "1" "2" "1" "1" ...
##  $ PROP_AGE       : chr  "103" "105" "104" "104" ...
##  $ PROP_TYPE      : chr  "106" "106" "106" "106" ...
##  $ FLOOR_AREA_BAND: chr  "1" "1" "1" "1" ...
##  $ EE_BAND        : chr  "2" "1" "3" "3" ...
##  $ LOFT_DEPTH     : chr  "99" "99" "99" "99" ...
##  $ WALL_CONS      : chr  "2" "1" "1" "1" ...
##  $ CWI            : chr  "" "" "" "" ...
##  $ CWI_YEAR       : chr  "" "" "" "" ...
##  $ LI             : chr  "" "" "" "" ...
##  $ LI_YEAR        : chr  "" "" "" "" ...
##  $ BOILER         : chr  "" "" "" "" ...
##  $ BOILER_YEAR    : chr  "" "" "" "" ...
```

- After performing the query, we can obtain additional information, for example its full SQL statement, the structure of the results set, and how many rows it returned:

```
info <- dbGetInfo(query.1)
str(info)

## List of 4
##  $ statement    : chr "SELECT * FROM need WHERE FLOOR_AREA_BAND = 1"
##  $ row.count    : int 50
##  $ rows.affected: int 0
##  $ has.completed: logi FALSE

info

## $statement
## [1] "SELECT * FROM need WHERE FLOOR_AREA_BAND = 1"
```

```
##
## $row.count
## [1] 50
##
## $rows.affected
## [1] 0
##
## $has.completed
## [1] FALSE
```

Once we complete a particular query, it is recommended to free the resources by clearing the obtained results set:

```
dbClearResult(query.1)
```

- We may now run a second query on the need table contained within the need_data SQLite database.
- This time, we will calculate the average electricity consumption for 2012 grouped by the levels of categorical variables: electricity efficiency band (EE_BAND), property age (PROP_AGE), and property type (PROP_TYPE).
- The statement will also sort the results set in ascending order, based on the values of two variables, EE_BAND and PROP_TYPE:

```
query.2 <- dbSendQuery(con, "SELECT EE_BAND, PROP_AGE, PROP_TYPE,
AVG(Econs2012) AS 'AVERAGE_ELEC_2012'
FROM need
GROUP BY EE_BAND, PROP_AGE, PROP_TYPE
ORDER BY EE_BAND, PROP_TYPE ASC")
```

- By running the statement, we simply apply a set of queries on the database; we then need to get the results into R using the fetch() function, just like we did in the first query. If you want to fetch all records, set the n parameter to -1:

```
query.2.res <- fetch(query.2, n=-1)
```

It's always a good idea to inspect the structure and the size of the results set created with the query:

```
info2 <- dbGetInfo(query.2)
info2

## $statement
## [1] "SELECT EE_BAND, PROP_AGE, PROP_TYPE,\nAVG(Econs2012) AS
'AVERAGE_ELEC_2012' \nFROM need\nGROUP BY EE_BAND, PROP_AGE, PROP_TYPE\nORDER
BY EE_BAND, PROP_TYPE ASC"
##
## $row.count
## [1] 208
##
## $rows.affected
## [1] 0
##
```

```
## $has.completed
## [1] TRUE
```

- From the preceding output, you can see that our results set consists of 208 rows of data with the variables as outlined in the fields attribute of the output. We can finally view the first six rows of data:

```
head(query.2.res, n=6)

##   EE_BAND PROP_AGE PROP_TYPE AVERAGE_ELEC_2012
## 1       1      101       101          2650.000
## 2       1      102       101         12162.500
## 3       1      103       101          3137.500
## 4       1      104       101          4200.000
## 5       1      105       101          3933.333
## 6       1      106       101          5246.774
```

- Before disconnecting from the database, you can also export the results set into a new table within the database:

```
dbWriteTable(con, "query_2_result", query.2.res, overwrite = T)

## Warning: Closing open result set, pending rows
```

- The new table named query_2_result has now been created in the need_data database:

```
dbListTables(con)

## [1] "myneed"         "need"           "query_2_result"
```

- Once you finish all the processing, make sure to clear the results of the most recent query and disconnect from the active connection:

```
dbClearResult(query.2)

## Warning: Expired, result set already closed

dbDisconnect(con)
```

- This completes the tutorial on SQLite database connectivity with the R language as a data source for locally run SQL queries.

- In the next section, we will explore how easily R can operate with a MariaDB database deployed on an Amazon EC2 instance.