Building MySQL R applications

Topic 4 Lesson 9

R connection to the client server model

As of today, there is no standard driver to connect an R program to a MySQL database

The DBI package separates the connectivity to the DBMS into a “front-end” and a “back-end”. Applications use only the exposed front-end API. The back-end facilities that communicate with specific DBMSs (SQLite, MySQL, PostgreSQL, MonetDB, etc.) are provided by drivers (other packages) that get invoked automatically through R’s S4 methods.

R to MySQL

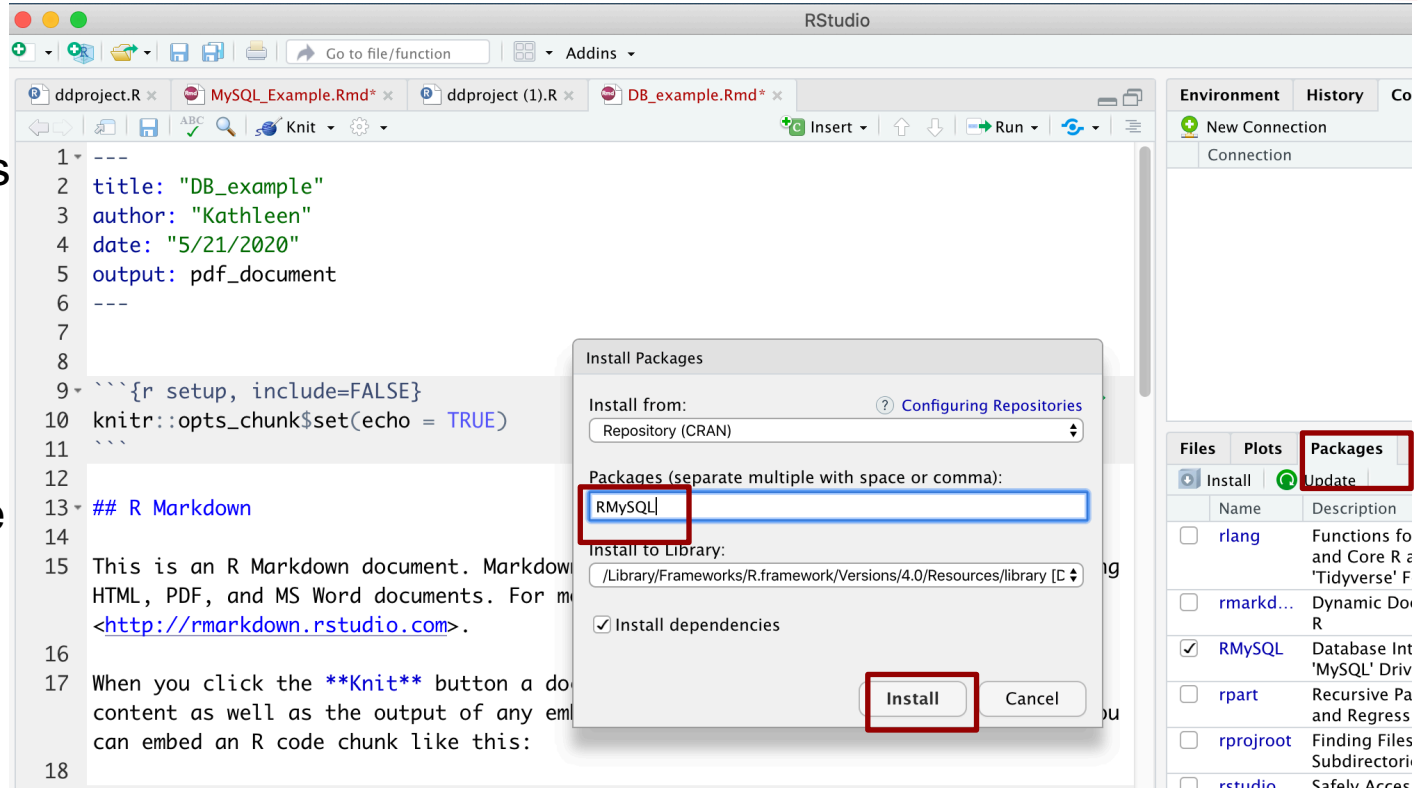
There are a few packages that do connect a R script to a MySQL database, RMySQL, RODBC, RMariaDB and RJDBC. (RMariaDB and RMySQL are supported) – both provide the same interface. All such libraries automatically include the DBI package.

Just like any other R package you must install the package locally so we can access its methods from our R script

Installing RMySQL

Issue the command: install.packages ("RMySQL") to install

You can also use the packages tab to install the library with the popup window



Connecting to the database

We use the dbConnect method to connect to the database. You need to provide your credentials, the hostname of the database and the port to connect. It returns an object with class MySQLConnection

EXAMPLE:

```
library(RMySQL)
mydb = dbConnect(MySQL(), user='user',
  password='password', dbname='database_name',
  host='host', port = 3306)
```

Documentation can be found:
https://cran.rproject.org/web/packages/RMySQL/RMySQL.pdf

Requesting and Accessing data

REQUEST data: dbSendQuery(con, sql) will retrieve data from the database a chunk at a time.

Parameters:

con is the value returned from DbConnect

sql is the query you wish to run on the database

Output: returns a MySQLConnection class

Accessing the requested data

ACCESS data: dbFetch(MySQLResult, n)

Parameters:

MySQLResult is the return variable from dbSendQuery

n maximum number of records to retrieve

Clean up: when done with the results free the allocated space with dbClearResult(res)

Requesting and Accessing data

REQUEST:dbGetQuery(con, sql) will retrieve data from the database

con parameter is the value returned from DbConnect

sql parameter is the query you wish to run on the database

It will automatically fetch all data locally and clear the space for the data. This should be used when the size of the returning data is small (will not exceed virtual memory of R program).

Example code (1)

```
library(RMySQL)
library(tidyverse)
globalUsername <- "root"
globalPass <- ”password"


# 1Settings
db_user <- 'root'
db_password <- ‘password'
db_name <- 'lotrfinal_1'

db_host <- '127.0.0.1' # for local access
db_port <- 3306

# 2. Connect to the db
mydb <-  dbConnect(MySQL(), user = db_user, password = db_password,
                   dbname = db_name, host = db_host, port = db_port)
```

Example code (2)

```
db_table <- 'lotr_character'

s <- str_c("select * from ", db_table)
# 2. Read from the db

rs <- dbSendQuery(mydb, s)
df <-  fetch(rs, n = -1)  #-1 represents to read all data
df
dbClearResult(rs)
dbDisconnect(mydb)
```

Example code (3)

```
# fetch chunks of data when dealing with large results

res <- dbSendQuery(con, "SELECT * FROM lotr_character")
while(!dbHasCompleted(res)){
  chunk <- dbFetch(res, n = 5)
  print(chunk)
  print("Next chunk")
  print(nrow(chunk))
}
dbClearResult(res)
```

Reading a table from the database

You can read a table from the database.

dbReadTable(con, name, row.names, check.names = TRUE, ...)

con – DBConnect object

name – name of the table

row.names - A string or an index specifying the column in the DBMS table to use as row.names in the output data.frame. Defaults to using the row_names column if present. Set to NULL to never use row names.

check.names - if TRUE, the default, column names will be converted to valid R identifiers

Writing a data frame to the database

You can write a data frame as a table to the database. This is useful for storing a data frame to permanent storage.

dbWriteTable(conn, name, value, field_types = NULL, row_names = TRUE, overwrite = FALSE, append = FALSE, ..., allow.keywords = FALSE)

con – DBConnect object
name – name of the table
value data frame to be stored as the table

Package RMariaDB

- For compatibility, the functions for connecting, retrieving and storing data in the database are the same as RMySQL.
- It also provides functions for managing transactions
- The data structures have different classes in the RMariaDB package the object returned from connect is a MariaDBConnection object
- MariaDBResult objects are returned from dbGetQuery.
- Please refer to the documentation:

https://cran.r-project.org/web/packages/RMariaDB/RMariaDB.pdf

A full tutorial can be found at https://programminghistorian.org/en/lessons/getting-started-with-mysql-using-r#selecting-data-from-a-table-with-sql-using-r

Summary

- R has the data frame object that an analogous to the structure of a relational table. We use a data frame object to accept data or pass data to/from the database.
- The MySQLConnection class is the object that tracks all information necessary for a connection (RMySQL)
- The MySQLResult class is the object that represents the data retrieved from the database (RMySQL)