Using R and MySQL

Ralph Mansson

Introduction

The R import/export manual discusses various approaches to handling data and mentions that R is not suitable for working with large data sets because data objects are stored in memory during a session. There are situations where using a database to hold the data and making use of one of the R libraries for database connectivity to access the data or to save the data.

MySQL is a popular database system and there is a library RMySQL that can be used to access this database it is important to ensure that the version of MySQL matches with the R library. If there is not a match then the system might exhibit erratic behaviour.

Connecting to a MySQL database

The first step is to make the RMySQL library available in the working session:

```
library(RMySQL)
```

If this command runs without problems then we need to create a connection object for our session:

```
con = dbConnect(MySQL(), dbname = "test",
user = "user1", password = "pwd1")
```

The first argument to this function creates and initializes a MySQL client. It returns an driver object that allows you to connect to one or several MySQL servers. The `dbname` argument is used to specify the name of the database and the user and password arguments should be self-explanatory.

After creating our connection successfully we can get R to list the tables that are stored in this database:

```
> dbListTables(con)
```

```
 1 "co2"
```

In this example there is only one table and its name is returned by the `dbListTables` function. To read the data from this table we use the `dbReadTable` function and specify the connection object as well as the name of the table in the MySQL database:

```
> dbReadTable(con, "co2")
```

```
 Plant Type Treatment conc uptake
1 Qn1 Quebec nonchilled 95 16.0
2 Qn1 Quebec nonchilled 175 30.4
3 Qn1 Quebec nonchilled 250 34.8
...
```

We can save the table to a data frame object rather than the default action of printing to the console.

After undertaken some analysis we might want to save a data set to the database and the dbWriteTable function is used:

```
> dbWriteTable(con, "CO2", data.frame(CO2),
overwrite = TRUE)
```

```
 1 TRUE
```

The first argument is the connection object, the second is the name that the table will be referred to in the database and the third argument is the data to be saved. In this case we have used the overwrite argument to copy over any existing table of the same name.

We can delete a table using the `dbRemoveTable` function:

```
> dbRemoveTable(con, "CO2")
1 TRUE
```

and when we reach the end of our need for the connection, the `dbDisconnect` function will remove the connection that we have been using:

```
> dbDisconnect(con)
1 TRUE
```

Running SQL Queries

We can run an SQL query with the dbGetQuery function which expects a connection object and a string with the SQL commands. If we wanted to take a subset of the CO2 dataset and view only those observations taken in Quebec we could use the following code:

```
> dbGetQuery(con, paste("select * from co2
where Type = 'Quebec'"))
```

```
 row_names Plant Type Treatment conc uptake
1 1 Qn1 Quebec nonchilled 95 16.0
2 2 Qn1 Quebec nonchilled 175 30.4
...
41 41 Qc3 Quebec chilled 675 39.6
42 42 Qc3 Quebec chilled 1000 41.4
```

We can narrow the search by specifying a condition on one of the other columns, for example limit the search to concentrations under 300 units:

```
> dbGetQuery(con, paste("select * from co2
where Type = 'Quebec' and Conc < 300"))
```

```
 row_names Plant Type Treatment conc uptake
1 1 Qn1 Quebec nonchilled 95 16.0
2 2 Qn1 Quebec nonchilled 175 30.4
...
17 37 Qc3 Quebec chilled 175 21.0
18 38 Qc3 Quebec chilled 250 38.1
```

As a final example we could look for all observations with a concentration less than 300 units buy order the data by increasing values of uptake:

```
> dbGetQuery(con, paste("select * from co2
where Conc < 300 order by uptake"))
```

```
 row_names Plant Type Treatment conc uptake
1 71 Mc2 Mississippi chilled 95 7.7
2 29 Qc2 Quebec chilled 95 9.3
...
35 38 Qc3 Quebec chilled 250 38.1
36 17 Qn3 Quebec nonchilled 250 40.3
```

The main advantage of this approach is that R does not have to try and store large datasets in memory which will cause the system to slow down or be unable to process the data if R cannot allocate sufficient resources.

GMR-2009-002: Using R and MySQL

This leaflet is part of a series of covering various topics involving the R Statistical Software.
http://www.gm-ram.com
http://www.wekaleamstudios.co.uk