Accessing Databases from R

Greater Boston useR Group
May 4, 2011

by

Jeffrey Breen
jbre@cambridge.aero
Outline

- Why relational databases?
- Introducing DBI
- Simple SQL queries
- `dbApply()` marries strengths of SQL and `*apply()`
- The parallel universe of RODBC
- `sqldf`: No database? No problem!
- Further Reading
- Loading mtcars sample data.frame into MySQL
Why relational databases?

- Databases excel at handling large amounts of, um, data
- They're everywhere
  - Virtually all enterprise applications are built on relational databases (CRM, ERP, HRIS, etc.)
  - Thanks to high quality open source databases (esp. MySQL and PostgreSQL), they're central to dynamic web development since beginning.
    - “LAMP” = Linux + Apache + MySQL + PHP
  - Amazon's “Relational Data Service” is just a tuned deployment of MySQL
- SQL provides almost-standard language to filter, aggregate, group, sort
  - SQL-like query languages showing up in new places (Hadoop Hive)
  - ODBC provides SQL interface to non-database data (Excel, CSV, text files)
Introducing DBI

- DBI provides a common interface for (most of) R's database packages
- Database-specific code implemented in sub-packages
  - RMySQL, RPostgreSQL, ROracle, RSQlite, RJDBC
- Use `dbConnect()`, `dbDisconnect()` to open, close connections:

```r
> library(RMySQL)
> con = dbConnect("MySQL", "testdb", username="testuser", password="testpass")
[...]
> dbDisconnect(con)
```
Using DBI

• dbReadTable() and dbWriteTable() read and write entire tables

```r
> df <- dbReadTable(con, 'motortrend')
> head(df, 4)

          mpg cyl disp  hp drat    wt qsec vs am gear carb     mfg      model
Mazda RX4  21.0   6 160 110 3.90 2.620 16.46  0  1    4    4   Mazda        RX4
Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02  0  1    4    4 Mazda RX4 Wag
Datsun 710  22.8   4 108  93 3.85 2.320 18.61  1  1    4    1 Datsun        710
Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1 Hornet 4 Drive
```

• dbGetQuery() runs SQL query and returns entire result set

```r
> df <- dbGetQuery(con, "SELECT * FROM motortrend")
> head(df,4)

          row_names mpg cyl disp  hp drat    wt qsec vs am gear carb    mfg   model
1      Mazda RX4 21.0   6 160 110 3.90 2.620 16.46  0  1    4    4  Mazda     RX4
2  Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02  0  1    4    4  Mazda RX4 Wag
3     Datsun 710 22.8   4 108  93 3.85 2.320 18.61  1  1    4    1 Datsun     710
4 Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1 Hornet 4 Drive
```

• Note how dbReadTable() uses “row_names” column

• Use dbSendQuery() & fetch() to stream larger result sets

• Advanced functions available to read schema definitions, handle transactions, call stored procedures, etc.
Simple SQL queries

Fetch a column with no filtering but de-dupe:

```r
> df = dbGetQuery(con, "SELECT DISTINCT mfg FROM motortrend")
> head(df, 3)
    mfg
1  Mazda
2  Datsun
3 Hornet
```

Aggregate and sort result:

```r
> df = dbGetQuery(con, "SELECT mfg, avg(hp) AS meanHP FROM motortrend GROUP BY mfg ORDER BY meanHP DESC")
> head(df, 4)
   mfg  meanHP
1 Maserati  335
2     Ford  264
3   Duster  245
4   Camaro  245
```

```r
> df = dbGetQuery(con, "SELECT cyl as cylinders, avg(hp) as meanHP FROM motortrend GROUP by cyl ORDER BY cyl")
> df
     cylinders  meanHP
1          4  82.63636
2          6 122.28571
3          8 209.21429
```
dbApply() marries strengths of SQL and *apply()

- Operates on result set from dbSendQuery()
  - Uses fetch() to bring in smaller chunks at a time to handle Big Data
  - You must order result set by your “chunking” variable

- Example: calculate quantiles for horsepower vs. cylinders

```r
sql = "SELECT cyl, hp FROM motortrend ORDER BY cyl"
rs = dbSendQuery(con, sql)
dbApply(rs, INDEX='cyl', FUN=function(x, grp) quantile(x$hp))
```

```r
$`4.000000`
  0%   25%   50%   75%   100%
  52.0  65.5  91.0  96.0 113.0

$`6.000000`
  0%   25%   50%   75%  100%
  105  110  110  123  175

$`8.000000`
  0%   25%   50%   75%   100%
  150.00 176.25 192.50 241.25 335.00
```

- Implemented and available in RMySQL, RPostgreSQL
The parallel universe of RODBC

- ODBC = “open database connectivity”
  - Released by Microsoft in 1992
  - Cross-platform, but strongest support on Windows
  - ODBC drivers are available for every database you can think of PLUS Excel spreadsheets, CSV text files, etc.
- For historical reasons, RODBC not part of DBI family
- Same idea, different details:
  - `odbcConnect()` instead of `dbConnection()`
  - `sqlFetch()` = `dbReadTable()`
  - `sqlSave()` = `dbWriteTable()`
  - `sqlQuery()` = `dbGetQuery()`
- Closest match in DBI family is RJDBC using Java JDBC drivers
sqldf: No database? No problem!

- Provides SQL access to data.frames as if they were tables
- Creates & updates SQLite databases automagically
  - But can also be used with existing SQLite, MySQL databases

```r
> library(sqldf)
> data(mtcars)
> sqldf("SELECT cyl, avg(hp) FROM mtcars GROUP BY cyl ORDER BY cyl")
   cyl avg(hp)
1   4  82.63636
2   6 122.28571
3   8 209.21429

> library(stringr)
> mtcars$mfg = str_split_fixed(rownames(mtcars), ' ', 2)[,1]
> sqldf("SELECT mfg, avg(hp) AS meanHP FROM mtcars GROUP BY mfg ORDER BY meanHP DESC LIMIT 4")
   mfg  meanHP
1 Maserati 335
2   Ford 264
3  Camaro 245
4  Duster 245
```
Further Reading

- Bell Labs: R/S-Database Interface

- R Data Import/Export manual
  - http://cran.r-project.org/doc/manuals/R-data.html#Relational-databases

- CRAN: DBI and “Reverse depends” friends
  - http://cran.r-project.org/web/packages/DBI/
  - http://cran.r-project.org/web/packages/RMySQL/
  - http://cran.r-project.org/web/packages/RPostgreSQL/
  - http://cran.r-project.org/web/packages/RJDBC/

- CRAN: RODBC
  - http://cran.r-project.org/web/packages/RODBC/

- CRAN: sqldf
  - http://cran.r-project.org/web/packages/sqldf/

- Phil Spector's SQL tutorial
In MySQL, create new database & user:

```sql
mysql> create database testdb;
mysql> grant all privileges on testdb.* to 'testuser'@'localhost' identified by 'testpass';
mysql> flush privileges;
```

In R, load "mtcars" data.frame, clean up, and write to new "motortrend" data base table:

```r
library(stringr)
library(RMySQL)

data(mtcars)

mtcars$mfg = str_split_fixed(rownames(mtcars), ' ', 2)[,1]
mtcars$mfg[mtcars$mfg=='Merc']= 'Mercedes'
mtcars$model = str_split_fixed(rownames(mtcars), ' ', 2)[,2]

con = dbConnect("MySQL", "testdb", username="testuser", password="testpass")

dbWriteTable(con, 'motortrend', mtcars)

dbDisconnect(con)
```