Connecting and Managing Databases with `aRT`

Pedro R. Andrade
Paulo Justiniano Ribeiro Junior

September 19, 2008

Contents

1 Introduction

`aRT` is an R package for manipulating spatial data using TerraLib. TerraLib manipulates data stored in DataBase Management Systems (DBMS), to which it is necessary to establish connections. Connecting to a DBMS is aways the first action when using `aRT`. This short document introduces the primary class of the package, named **aRTconn**. This class implements a virtual connection to a DBMS. In the initial examples, we consider the user already has permissions in the DBMS. Later, in Section 4, we explain how `aRT` functions can change the permissions in the DBMS.

```
> require(aRT)
> aRTsilent(FALSE)

[1] FALSE
```

2 Connections

After loading the package, it is necessary to establish a DBMS connection. The function `openConn()` is designed for such task. Currently, this function takes four arguments (the defaults values are indicated within parenthesis): `user` (the current user as given by $USER), `password` (no password), `host` ('localhost'), and `port` (3306). Future versions will include a fifth argument, `DBMS`, with options to supported BDMS such as "MySQL", "Postgres", "PostGIS".

```
> con = openConn(user="root", host="localhost", pass="")
> con
```

When a connection is successfully stablished, this function returns an object of the class **aRTconn**. It is important to notice that the elements in the object **con** cannot be changed. The only possible way to set/change the connection parameters is creating another object. calling **openConn()** with the new options. This happens because its data is stored in an external pointer. If the connection cannot be established, the function stops with an error, as in the following example:

```
> err= try(conn2 <- openConn(user="root", pass="abc321"))

Trying to connect ... no

> strsplit(err[1]," : ")

[[1]]
[1] "Error in .local(.Object, ...)"
[2] "\n  Access denied for user 'root'@'localhost' (using password: YES)\n"
```

An **aRTconn** object stores a *virtual* connection to a DBMS, that is, every time a database access is required, the object connects with the DBMS, performs the task, and then disconnects. **aRTconn** objects allow for some basic queries and operations in the DBMS. The function **showDbs()** lists the databases available (the ones which the user has some permission):

```
> showDbs(con)

 [1] "information_schema" "auckland"           "bodmin"
 [4] "ca20"               "catarina"           "gpm"
 [7] "hello"              "mysql"              "parana"
[10] "performance_schema" "pol3"               "recife"
[13] "rondonia"           "sp"                 "tabletest"
[16] "test"
```

New databases can be created with **createDb()**:

```
> dbintro = createDb(con, "intro")

Creating database 'intro' ... yes
Creating conceptual model of database 'intro' ... yes
Creating application theme table 'intro' ... yes
Loading layer set from database 'intro' ... yes
Loading view set 'root' from database 'intro' ... yes

> dbintro
```

```
Object of class aRTdb

Database: "intro"
Layers: (none)
Themes: (none)
External tables: (none)
```

and existing databases can be opened with openDb(),

```
> db = openDb(con, "intro")

Connecting to database 'intro' ... yes
Connected to database version 4.1.2
Loading layer set from database 'intro' ... yes
Loading view set 'root' from database 'intro' ... yes

> db

Object of class aRTdb

Database: "intro"
Layers: (none)
Themes: (none)
External tables: (none)
```

In these examples, both dbintro and db belong to the class aRTdb, that stores a real connection to a particular database. As a consequence, at this point the virtual connection is no longer needed and these objects are independent from con. Note that an aRTdb object can turn inconsistent if the database is removed from the DBMS, possibly generating a *core dump*. This cannot be avoided because the database can be removed from any other connection to the DBMS, which can be from another aRTconn object, directly from MySQL, or by another TerraLib-based application.

3 Deleting databases

Whenever the user removes the object that stores a database connection, the connection is not removed yet. To successfuly remove a database connection, you have to to call the garbage collector using gc().

```
> rm(db)
> rm(dbintro)
> invisible(gc())

Removing aRTdb 'intro' from memory ... yes
Removing aRTdb 'intro' from memory ... yes
```

To physically remove a database, the user needs a special permission. If you have such permission, it is possible to remove databases using `deleteDb()`. The argument `force=TRUE` can be used to avoid a keyboard confirmation, because this a dangerous operation, and cannot be undone.

```
> if(any(showDbs(con) == "intro")) deleteDb(con, "intro", force=TRUE)
```

```
Checking for database 'intro' ... yes
Deleting database 'intro' ... yes
```

4   Managing users from aRT

The function `addPermission()` provides a way to add users to the DBMS, with corresponding permissions. The first step is to start a session with *root* permissions.

```
> con=openConn(u="root")
```

Using this conection, some types of permissions can be set[1]:

- To create an user with access from localhost, without or with password use, respectively:

  ```
  > addPermission(con, "elias")
  > addPermission(con, "elias", pass="password")
  ```

- To create an user with access from a specific host, specified either by an IP number or hostname, without or with password use, respectively:

  ```
  > addPermission(con, "elias", host="est.ufmg.br")
  > addPermission(con, "elias", host="est.ufmg.br", pass="senha")
  ```

- Finally, to create an user from any host the password is compulsory:

  ```
  > addPermission(con, "elias", remote=TRUE, pass="senha")
  ```

This function, when called as before, provides **full access** to all the DBMS databases. It is also possible to restrict the permissions to some privileges or some specific databases. For instance:

```
> addPermission(con, "elias", db="citrus")
> addPermission(con, "elias", privilege="select", db="saudavel")
> addPermission(con, "elias", privilege="update", db="parana")
```

In order to see the permissions of the DBMS, use `getPermissions`:

```
> getPermissions(con)[1:5,]
```

---

[1]Henceforth, the R code will not be executed for security reasons.

```
                   host user password select insert update
1                localhost root      Yes    Yes    Yes    Yes
2 maquinavirtual-VirtualBox root      Yes    Yes    Yes    Yes
3                127.0.0.1 root      Yes    Yes    Yes    Yes
4                      ::1 root      Yes    Yes    Yes    Yes
5                localhost           No     No     No     No
  delete create
1    Yes    Yes
2    Yes    Yes
3    Yes    Yes
4    Yes    Yes
5     No     No
```

Finally, for removing the permissions of a given user, use `dropUser`:

```
> dropUser(con, user="elias")
> dropUser(con, user="elias", remote=TRUE)
> dropUser(con, user="elias", host="pataxo.est.ufpr.br")
```

References

[1] Chambers, J.M., 1998, Programming with data, a guide to the S language. Springer, New York.

[2] MySQL Documentation. `http://dev.mysql.com/doc/`. Last access: 08/14/2008