

Package ‘SQLiteMap’

April 17, 2009

Type Package

Title package to manage vector graphical maps using SQLite

Version 0.3

Date 2008-07-22

Author Norbert Solymosi

Maintainer Norbert Solymosi <solymosi.norbert@gmail.com>

Depends R (>= 2.7.0), RSQLite, sp, maptools

Description Some server based database management systems implemented the OpenGIS Simple Features Specification for SQL. The OpenGIS specification defines two standard ways of expressing spatial features, the Well-Known Text (WKT) form and the Well-Known Binary (WKB) form. Both WKT and WKB include information about the type of the feature and the coordinates which form the feature. These systems (e.g. PostgreSQLPostGIS, MySQL, ORACLE, MSSQL) allow to store the topological features and the descriptive data in the same database. But these systems assume that the user needs permission to a running service or to install a server to use the spatial data. In some cases, it is useful if the user can use the database stored maps on different computers and platforms. The SQLite is a good choice for a portable database, it is platform-independent and there are some R packages to manage SQLite databases. Unfortunately, it has no spatial extension, but there is an SQLite extension for the SharpMap library. Following the idea of this solution this package was developed that may help the user read and write spatial features from and to an SQLite database. Each table with geometry field is treated as a layer. The tables contain the topological features in one geometry field in WKT form. For conversion of standard map files to SQLite table maps2WinBUGS can be used.

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2008-07-29 04:59:22

R topics documented:

SQLiteMap-package	2
sqli.dump	3
sqli2map	4
sqli2sp	5

Index	7
--------------	----------

SQLiteMap-package *A package to manage vector graphical maps using SQLite*

Description

Some server based database management systems implemented the OpenGIS "Simple Features Specification for SQL". The OpenGIS specification defines two standard ways of expressing spatial features: the Well-Known Text (WKT) form and the Well-Known Binary (WKB) form. Both WKT and WKB include information about the type of the feature and the coordinates which form the feature. These systems (e.g. PostgreSQLPostGIS, MySQL, ORACLE, MSSQL) allow to store the topological features and the descriptive data in the same database.

But these systems assume that the user needs permission to a running service or to install a server to use the spatial data. In some cases, it is useful if the user can use the database stored maps on different computers and platforms. The SQLite is a good choice for a portable database, it is platform-independent and there are some R packages to manage SQLite databases. Unfortunately, it has no spatial extension, but there is an SQLite extension for the SharpMap library.

Following the idea of this solution this package was developed that may help the user read and write spatial features from and to an SQLite database. Each table with geometry field is treated as a layer. The tables contain the topological features in one geometry field in WKT form.

For conversion of standard map files to SQLite table maps2WinBUGS can be used.

Details

Package:	SQLiteMap
Type:	Package
Version:	0.3
Date:	2008-07-28
License:	GPL-2
LazyLoad:	yes

Author(s)

Norbert Solymosi

Maintainer: Norbert Solymosi <solymosi.norbert@gmail.com>

References

<http://www.opengeospatial.org/standards>
<http://postgis.refractions.net/>
<http://www.codeplex.com/SharpMap>
<http://sourceforge.net/projects/maps2winbugs>

sqli.dump	<i>Save maps into SQLite database</i>
-----------	---------------------------------------

Description

Save Map or Spatial objects into SQLite database in two geometry and attribute tables.

Usage

```
sqli.dump(db, mapobj, mn)
```

Arguments

db	path of SQLite database
mapobj	the Map or Spatial object
mn	save as name of object

Author(s)

Norbert Solymosi <solymosi.norbert@gmail.com>

See Also

[sqli2map](#), [sqli2sp](#), [maptools](#) package, [sp](#) package

Examples

```

sqli.db <- system.file("sqlimaps/sids.db3", package="SQLiteMap")
drv <- dbDriver("SQLite")
con <- dbConnect(drv, dbname = sqli.db)

sql <- 'select sidsmap.gid, sidsmap.geom, sidsattr.*
      from sidsmap Inner Join sidsattr On sidsattr.sp_id = sidsmap.sp_id
      order by sidsattr.name'
rs <- dbSendQuery(con, sql)
join.data <- fetch(rs, n = -1)

sids.sp <- sqli2sp(geoms=join.data, gcol='geom', idcol='name')
sids.attr <- data.frame(R74 = join.data$sid74/join.data$bir74,
                      R79 = join.data$sid79/join.data$bir79)
rownames(sids.attr) <- join.data$name
sids.df <- SpatialPolygonsDataFrame(sids.sp, sids.attr)

```

```
sqli.dump(db = 'test.db3', mapobj = sids.df, mn = 'sidsexport')
```

sqli2map

Read SQLite geometry table into Map object

Description

From SQLite database transforms the source geometries in WKT form to Map object.

Usage

```
sqli2map(geoms, gcol)
```

Arguments

geoms	table contains the WKT geometry field
gcol	the geometry field of geoms table

Value

Returns Map object depending on the source geometry type.

Author(s)

Norbert Solymosi <solymosi.norbert@gmail.com>

See Also

[sqli2sp](#), [sqli.dump](#), [maptools](#) package

Examples

```
sqli.db <- system.file("sqlimaps/sids.db3", package="SQLiteMap")
drv <- dbDriver("SQLite")
con <- dbConnect(drv, dbname = sqli.db)

sql <- 'select * from sidsmap order by gid'
rs <- dbSendQuery(con, sql)
geom.tab <- fetch(rs, n = -1)

sids.map <- sqli2map(geoms=geom.tab, gcol='geom')
```

`sqli2sp`*Read SQLite geometry table into spatial*

Description

From SQLite database transforms the source geometries in WKT form to SpatialPolygons, SpatialLines or SpatialPoints object.

Usage

```
sqli2sp(geoms, gcol, idcol)
```

Arguments

<code>geoms</code>	table contains the WKT geometry field
<code>gcol</code>	the geometry field of <code>geoms</code> table
<code>idcol</code>	field of <code>geoms</code> table for identification

Value

Returns SpatialPolygons, SpatialLines or SpatialPoints depending on the source geometry type.

Author(s)

Norbert Solymosi <solymosi.norbert@gmail.com>

See Also

[sqli2map](#), [sqli.dump](#), `sp` package

Examples

```
sqli.db <- system.file("sqlimaps/sids.db3", package="SQLiteMap")
drv <- dbDriver("SQLite")
con <- dbConnect(drv, dbname = sqli.db)

sql <- 'select sidsmap.gid, sidsmap.geom, sidsattr.*
      from sidsmap Inner Join sidsattr On sidsattr.sp_id = sidsmap.sp_id
      order by sidsattr.name'
rs <- dbSendQuery(con, sql)
join.data <- fetch(rs, n = -1)

sids.sp <- sqli2sp(geoms=join.data, gcol='geom', idcol='name')
sids.attr <- data.frame(R74 = join.data$sid74/join.data$bir74,
                      R79 = join.data$sid79/join.data$bir79)
rownames(sids.attr) <- join.data$name
sids.df <- SpatialPolygonsDataFrame(sids.sp, sids.attr)

library(RColorBrewer)
```

```
splot(sids.df,  
      col.regions = colorRampPalette(brewer.pal(9, "OrRd"))(140))
```

Index

*Topic **spatial**

- `sqli.dump`, [3](#)
- `sqli2map`, [4](#)
- `sqli2sp`, [5](#)
- `SQLiteMap-package`, [2](#)

- `sqli.dump`, [3](#), [4](#), [5](#)
- `sqli2map`, [3](#), [4](#), [5](#)
- `sqli2sp`, [3](#), [4](#), [5](#)
- `SQLiteMap` (*SQLiteMap-package*), [2](#)
- `SQLiteMap-package`, [2](#)