SQLiteMap: package to manage vector graphical maps using SQLite

Norbert Solymosi,[1] Andrea Harnos,[1,2] Jenő Reiczigel[1,2]

[1]Adaptation to Climate Change Research Group, Hungarian Academy of Science
Budapest, Hungary

[2]Department of Biomathematics and Informatics, Faculty of Veterinary Science
Szent István University, Budapest, Hungary

The R User Conference 2008
August 12-14, Technische Universität Dortmund, Germany

GIS – Maps

- File based standards:
  - ESRI Shape, MapInfo TAB, ...
  - Difficulties in handling attribute data
- Spatial database:
  - The Open Geospatial Consortium (OGC) created the Simple Features specification and sets standards for adding spatial functionality to database systems.
  - PostgreSQL–PostGIS, MySQL, ...
  - OGC defines two standard ways of expressing spatial objects:
    - Well-Known Text (WKT)
    - Well-Known Binary (WKB)
  - Server based solutions assume that the user needs permission to a running service or to install a server to use the spatial data.

SQLite database

"Small. Fast. Reliable. Choose any three."

- Free, cross-platform, no configuration
- Growing usage in R
  - `RSQLite`
  - `filehashSQLite`
  - `SQLiteDF`
  - `TSSQLite`
  - Bioconductor:
    - e.g. Annotation, CDF and Probe packages

SQLiteMap

- An R-interface between SQLite and Map or Spatial objects
- SharpMap library approach `http://www.codeplex.com/SharpMap`

Spatial table

- Join field
- Geometry field (WKT)
- `minx`, `miny`
- `maxx`, `maxy`
- ...

Attribute table(s)

- Join field
- ...

Dependencies

- `RSQLite`
- `sp`
- `maptools`

WKT

```
POINT(0 0)
LINESTRING(0 0,1 1,1 2)
POLYGON(
        (0 0,4 0,4 4,0 4,0 0),
        (1 1, 2 1, 2 2, 1 2,1 1)
        )
MULTIPOINT(0 0,1 2)
MULTILINESTRING(
        (0 0,1 1,1 2),
        (2 3,3 2,5 4)
        )
MULTIPOLYGON(
        ((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)),
        ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1))
        )
```

sqli2map()

```
> sqli.db <- 'maps.db3'
> drv <- dbDriver('SQLite')
> con <- dbConnect(drv, dbname = sqli.db)
> sql <- 'select * from choros order by gid'
> rs <- dbSendQuery(con, sql)
> geom.tab <- fetch(rs, n = -1)
```

Data Editor

File Edit Help

	gid	geom	minx	miny	maxx	maxy
1	1	POLYGON ((597142.9 85527.101, 597905.8 >	585128.9	70327.8	603330.9	92222.7
2	2	POLYGON ((584474.902 89804.801, 585316.>	573280.8	78858.2	586880.7	95662.4
3	3	POLYGON ((583577.497 79681.437, 583652.>	576993.8	65126.9	590715.4	81376.12
4	4	POLYGON ((581501.402 108766.999, 581417>	563083.3	86563.5	601837.9	120160.4
5	5	MULTIPOLYGON (((621628.098 87923.201, 6>	596341.6	52481	635537	107161
6	6	POLYGON ((557287.099 50203.8, 555613.29>	552843	44176.3	613527.8	75726.2
7	7	POLYGON ((563480.499 96418.7, 564435.59>	540554.2	53109.5	579271.7	100109.5

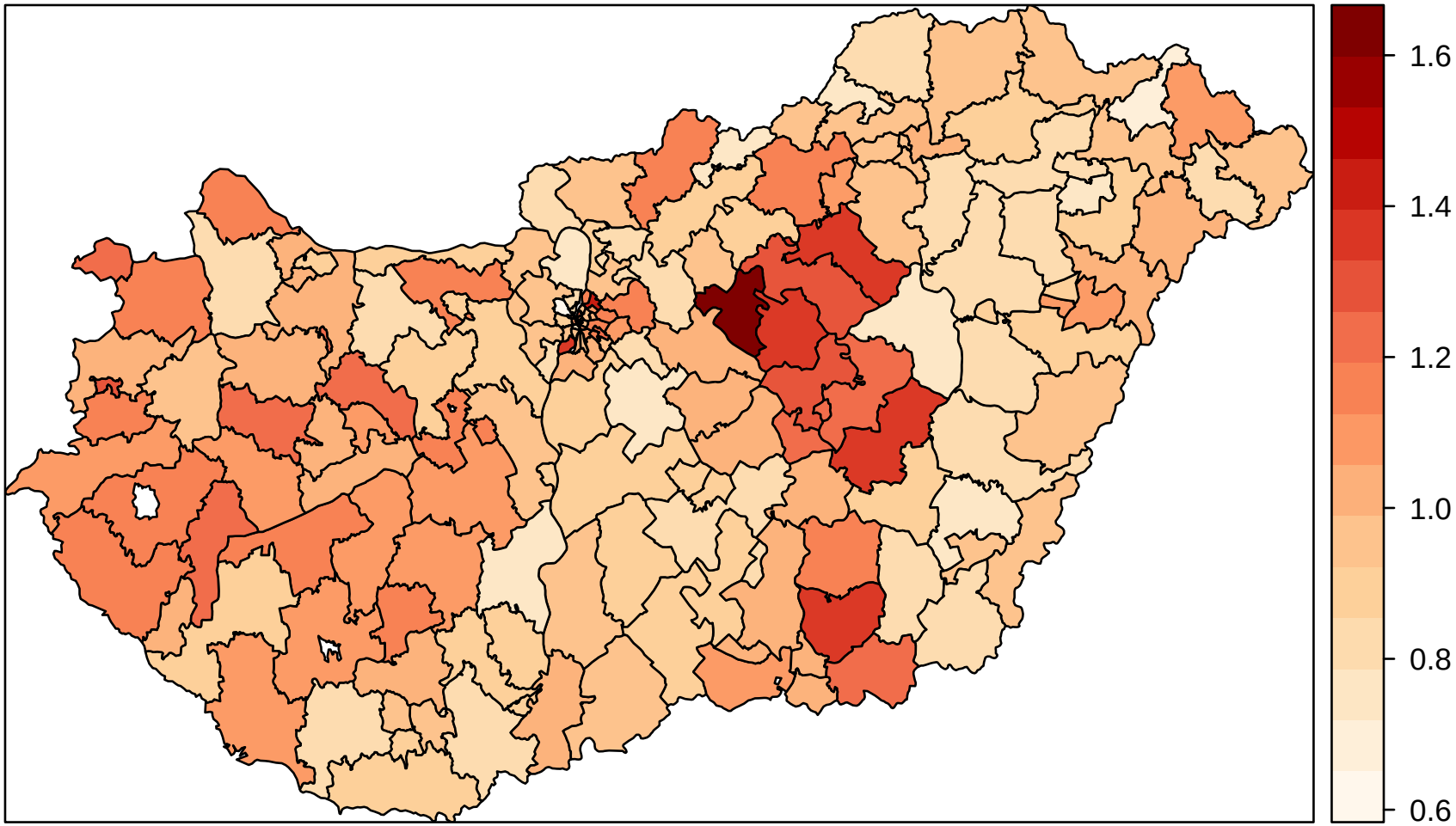
```
> choros.map <- sqli2map(geoms=geom.tab, # spatial table
+ gcol='geom')                          # WKT geometry field
```

sqli2sp()

```
> sqli.db <- 'maps.db3'
> drv <- dbDriver('SQLite')
> con <- dbConnect(drv, dbname = sqli.db)
> sql <- 'select choros.*, rrtab.*
+ from choros Inner Join rrtab On rrtab.gid = choros.gid
+ order by choros.gid'
> rs <- dbSendQuery(con, sql)
> join.data <- fetch(rs, n = -1)

> choros.sp <- sqli2sp(geoms=join.data, # spatial table
+ gcol='geom',                         # WKT geometry field
+ idcol='gid')                         # identification field

> choros.attr <- data.frame(RR90 = join.data$RR90)

> rownames(choros.attr) <- join.data$gid
> choros.df <- SpatialPolygonsDataFrame(choros.sp, choros.attr)
```

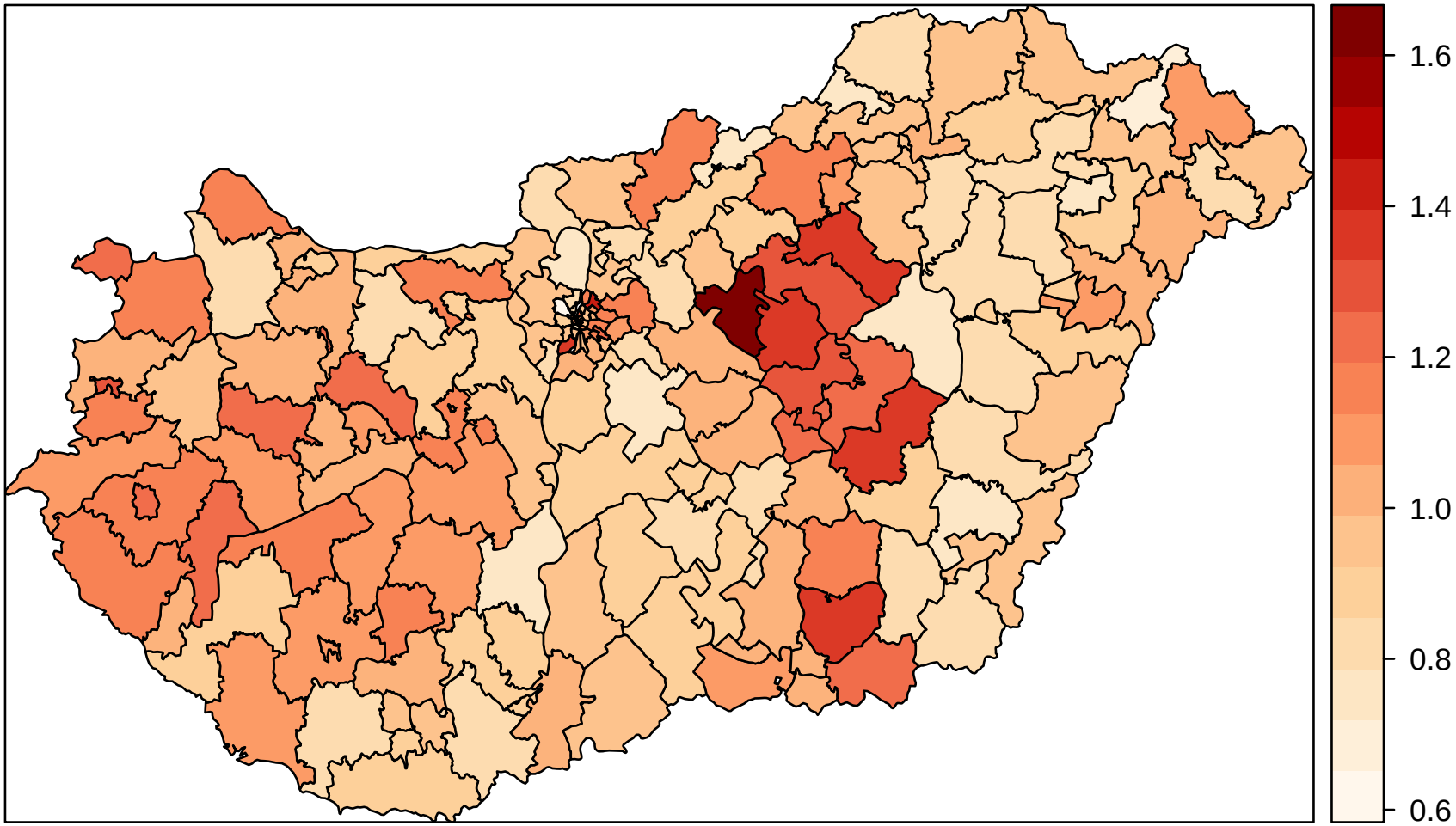
sqli2sp()

```
> library(RColorBrewer)
> spplot(choros.df,
+ col.regions = colorRampPalette(brewer.pal(9,'OrRd')[1:9])(140))
```



spdfho()

```
> choros.df2 = spdfho(choros.df)
> spplot(choros.df2,
+ col.regions = colorRampPalette(brewer.pal(9,'OrRd')[1:9])(140))
```



sqli.dump()

```
> sqli.db <- 'maps.db3'
> drv <- dbDriver('SQLite')
> con <- dbConnect(drv, dbname = sqli.db)
> sql <- 'select choros.*, rrtab.*
+ from choros Inner Join rrtab On rrtab.gid = choros.gid
+ order by choros.gid'
> rs <- dbSendQuery(con, sql)
> join.data <- fetch(rs, n = -1)
> choros.sp <- sqli2sp(geoms=join.data, gcol='geom', idcol='gid')
> choros.attr <- data.frame(RR90 = join.data$RR90,
+ RR98 = join.data$RR98)
> rownames(choros.attr) <- join.data$gid
> choros.df <- SpatialPolygonsDataFrame(choros.sp, choros.attr)

> sqli.dump(db = 'test.db3',      # path of SQLite database
+ mapobj = choros.df,             # Map or Spatial object
+ mn = 'chorosexport')            # save as name of object
```

Dump the Map or Spatial object into SQLite database

- `chorosexport` spatial table
- `chorosexportdt` attribute table

Acknowledgments

Ferenc Péter Speiser

Supported by OTKA T049157

Acknowledgments

Ferenc Péter Speiser

Supported by OTKA T049157

Thanks for Your Attention!

`http://cran.r-project.org/web/packages/SQLiteMap`