Geospatial Data in R

Going off-road

Barry Rowlingson
b.rowlingson@lancaster.ac.uk

School of Health and Medicine,
Lancaster University
Spatial Packages

What's on CRAN for us?
Spatial Task View

Spatial Task View is a framework for analyzing spatial data. This framework provides a comprehensive approach to understanding and analyzing spatial relationships and patterns in data. It includes tools and methods for data visualization, data modeling, and spatial analysis.

The Spatial Task View framework consists of several key components:

1. **Data Preparation**: This involves the collection, cleaning, and preprocessing of spatial data. This step is crucial to ensure that the data is suitable for analysis.
2. **Data Visualization**: Tools and techniques for visualizing spatial data, including maps and charts, to help in understanding spatial patterns and relationships.
3. **Data Modeling**: Techniques for modeling spatial relationships, such as regression analysis, to predict and understand spatial trends.
4. **Spatial Analysis**: Methods for analyzing spatial data, including spatial clustering, spatial regression, and spatial autocorrelation.
5. **Spatial Prediction**: Tools for predicting spatial outcomes based on spatial data.
6. **Spatial Decision Making**: Techniques for making informed decisions based on spatial analysis.

The Spatial Task View framework is designed to be flexible and scalable, allowing for the integration of various spatial data and analysis tools. It is a valuable resource for researchers, planners, and policymakers who need to analyze and understand spatial data.

---

**Key Features**

- **Comprehensive Approach**: The Spatial Task View framework provides a holistic approach to spatial data analysis, covering data preparation, data visualization, data modeling, spatial analysis, spatial prediction, and spatial decision making.
- **Flexible and Scalable**: The framework is designed to be flexible and scalable, allowing for the integration of various spatial data and analysis tools.
- **User-Friendly**: The framework includes user-friendly tools and interfaces to facilitate the analysis of spatial data.

---

**Spatial Task View is a valuable resource for researchers, planners, and policymakers who need to analyze and understand spatial data.**
1 Package
73 Packages
116 Packages
122 Packages
A package does what?

- Geography/Geometry
  - `gdistance`, `geosphere`, `voronoi`, ...
- General Spatial Statistics
  - `geoR`, `spatstat`, `lgcp`, `spgwr`, `gstat`, `stam`, ...
- Specific Statistics
  - `adeHabitat*`, `Rquake`, `fossil`, ...
- Data
  - `UScensus*`, `MUCflights`, `RghcnV3`, ...
Sounds like a lot of R

- Over 3000 packages on CRAN
- Must be millions of lines of code
- It's not all R
Time travel

Historical Perspective
Scientists and Statisticians wrote FORTRAN and C code

Everybody started pretty much from scratch

We realised we didn't all have to reinvent the wheel

Useful modules of code got arranged into neat, re-usuable libraries

Scientists linked their code with the libraries.
But...

- The process was slow...
  - Write FORTRAN or C code
  - Compile
  - Link with libraries
  - Run
  - Print output
Graphs

The image contains a table with values ranging from 0.00 to 3.14. Each value is paired with a corresponding symbol (*).
Or plot

- On a graphics terminal
Almost as good...
Anyway...

- Graphics Improved
- Communications Improved
- Processors Got Faster
- Statisticians and Scientists Got Impatient
- Started using S
- Interactive, responsive, data handling, graphics
- Perfect Storm for data scientists
What about all my C and FORTRAN code?

mypi.c

```c
void value(double &p){
    &p = 3.14;
    return;
}
```

mypi.so

```r
mypi = function(){
    .C('value',0.0)[1]
}
```

```r
mypi()
[1] 3.14
```
Entire wrapped C libraries

- **GEOS**
  - Geometry Engine Open Source
  - Wrapped by `rgeos`

- **PROJ4**
  - Cartographic Projections
  - Called by `sp:spTransform` and `raster:project`

- **GDAL/OGR**
  - Read raster and vector datasets
  - Wrapped by `rgdal`, used by `raster`, `sp`
OSGeo Projects

- Open Source Geospatial Foundation

- Supports development of geospatial software

- Many of which play nicely with R
Quantum GIS

- Desktop GIS
- Windows
- Linux
- Mac
- Written in C++
- Embedded Python
- Extensible in C++ and Python
What do I use it for?

- Interactive Mapping
- Cartography
- Working with Databases
Qgis-R Integration

package:rgdal
or
package:raster
1 \text{z} = \text{function}(x)[\sqrt{x}]

Histogram of values(cumbria)
Qgis Plugins with R

- Qgis Python Plugin
  - Adds functionality to Qgis
- Python plugin calls R
  - Python Rpy2 module
- R gets data from Qgis. R writes data to files
- Python plugin adds data to Qgis
PostGIS

- Spatially Enables the PostgreSQL DB
- Adds Geometry Columns to Databases
- Provides Spatial Queries
- Client: either Rpgsql or rgdal
- Server: PL/R
What do I use it for?

- Managing big data sets
- Spatial queries
Databases

- SQL Queries
  - SELECT * from employees where salary > 10000

- Spatial SQL Queries
  - SELECT ST_Buffer(the_geom, range) as the_geom from phone_masts;
R with PostGIS

- Via RpostgreSQL

```r
> con = dbConnect(dbDriver('PostgreSQL'),
    user='gis', dbname='maps')
> rs = dbSendQuery(con,
    'select * from countries where pop > 1000000')
> countries = fetch(rs, -1)
```
R with PostGIS

- Via `rgdal`

```r
> countries = readOGR('PG: dbname=maps', 'countries')
```
PostGis Integration

- GDAL
- package:rgdal
- PostGIS db
- Spatial tables
PL/R architecture

Can add extra functionality – even spatial functionality – to the database
Get mean age of people in each town:

```sql
SELECT town, mean(age) AS m FROM people
    GROUP BY town ORDER BY m;
```

No 'median' function in SQL- so we'll make one...

```sql
CREATE function r_median(_float8)
    RETURNS FLOAT AS 'median(arg1)' LANGUAGE 'plr';
```

```sql
CREATE AGGREGATE median (    
    sfunc = plr_array_accum,    
    basetype = float8,    
    stype = _float8,    
    finalfunc = r_median
);
```

```sql
SELECT town, median(age) AS m FROM people
    GROUP BY town ORDER BY m;
```
webmaps package (on r-forge)

```r
> settlements = readOGR(Datadir,"settlements.shp")
> slayer = layer(spTransform(settlements,
    CRS('+init=epsg:4326')), 'settlements')
> mway = readOGR(Datadir,"mways.shp")
> mlayer = layer(spTransform(mway,
    CRS('+init=epsg:4326')), "Motorways")
> osmMap(slayer,mlayer,title="Towns and Roads")
[1] "/tmp/Rtmbhf7sdd/index.html"
```
Open...

- Fully working interactive web-based map
- Put on a server, share with world
- Customise HTML and JS to suit
function init()
{
    var options = {
        projection: new OpenLayers.Projection("EPSG:900913"),
        displayProjection: new OpenLayers.Projection("EPSG:4326"),
        units: "m",
        maxResolution: 156543.0339,
        maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
                                         20037508.34, 20037508.34),
        controls: [new OpenLayers.Control.Navigation(),
                    new OpenLayers.Control.PanZoomBar(),
                    new OpenLayers.Control.LayerSwitcher(),
                    new OpenLayers.Control.Permalink(),
                    new OpenLayers.Control.ScaleLine(),
                    new OpenLayers.Control.mousePosition()]
    };

    map = new OpenLayers.Map("map", options);
    var mapnik = new OpenLayers.Layer.OSM.Mapnik("OpenStreetMap");
    map.addLayer(mapnik);

    var settlements = new OpenLayers.Layer.GML(
        "settlements","settlements.gml",
        {projection: new OpenLayers.Projection("EPSG:4326")}
    );

    map.addLayer(settlements);
webmaps also does tiles

```r
cumbriaT = getTiles(c(-3.253926, -2.796769), c( 54.255070, 54.524853), 11, path="http://tile.openstreetmap.org/")

SLL = spTransform(settlements, CRS('+init=epsg:4326'))

image(cumbriaT)
points(SLL)
title('settlements')
```

Grabs tiles off servers for plotting in R
Other packages that can do this...

```r
> library(OpenStreetMap)
> map = openmap(c(54.524, -3.523), c(54.255, -2.797), type="osm")
> plot(map)
```

Uses Java, heavyweight
ggmap

```r
> gm = get_map(location=c(-3.553, 54.155, -2.796, 54.524))
> gm2 = get_map(location=c(-3.553, 54.155, -2.796, 54.524),
                 source="stamen", maptype="toner")
> ggmap(gm)
> ggmap(gm2)
```

[I have figured out how to get this into a raster format]
ggmap with data

ggmap(gm) +
geom_point(
  aes(
    x=Easting, y=Northing,
    col=Crime.type),
  data=scD)
ggmap with polygons

```
ggmap(gm) + geom_polygon(
  aes(fill=Population,
      group=id,
      x=long, y=lat),
  data=datapoly, alpha=0.7) +
  scale_fill_continuous(
    low="white", high="red")
```

I couldn't get `geom_map` to work with `ggmap`
Raw OSM Data
But what about Google Earth?

- Isn't it lovely?
- Look, I can spin and zoom!
- And overlay!
- And animate!

BUT

- As I keep telling people...
ITS NOT OPEN SOURCE!
However...

KML is an OGC standard
All 233 pages of it
Making KML

Vectors

> `writeOGR(Towns, "towns.kml", "towns", "KML")`
Making KML

Rasters
> `demLL = projectRaster(dem, crs = "+init=epsg:4326")`
> `KML(demLL, "dem.kmz")`
<Placemark>
  <ExtendedData><SchemaData schemaUrl="#towns">
    <SimpleData name="pop">18599</SimpleData>
  </SchemaData></ExtendedData>
  <Point>
    <coordinates>
      -1.248,54.09701
    </coordinates>
  </Point>
</Placemark>

Using the **brew** package, write a template. Here's an extract:

```html
<% for(pt in points){ %>
<Placemark>
  <ExtendedData><SchemaData schemaUrl="#towns">
    <SimpleData name="pop"><%= pt$pop %></SimpleData>
  </SchemaData></ExtendedData>
  <Point>
    <coordinates>
      <%=pt$lon%>,<%=pt$lat%>
    </coordinates>
  </Point>
</Placemark>
<% } %>
```
Spatial Data Infrastructure

GeoFOSS based SDI Software Architecture

Client applications
- Thick (Desktop) Clients (uDig, QGIS, gvSIG, ArcGIS, Google Earth)
- Thin (Web) Clients (MapBuilder, InterMap, ka-Map, OpenLayers)

Interfaces
- Direct Access
- WMS (GIF, PNG, JPG, KML)
- (T-)WFS (GML, Shapefile)
- WCS (TIFF, GeoTIFF, ...)
- CSW (DC, FGDC, ISO19115/19139)

Servers
- Map Servers (GeoServer, MapServer, Deegree)
- Catalog Servers (GeoNetwork)

Databases
- (Geo-)Databases (MySQL, PostgreSQL/PostGIS DBMS)
- (Vector data (both archive and for editing), Metadata)

File System
- (Vector data & satellite images)

GeoNetwork opensource 2006 - v2
Spatial Data Infrastructure

Java
C/C++
R
Python
Jscript
R
Python
C/C++
Python
Java
C/C++
Python
R
C/C++
Python
R
C/C++
Python
R
C/C++
Python
R
C/C++
Python
R
C/C++

GeoNetwork opensource 2006 - v2

ICAS combining health information, computation and statistics
FOSS4G 2013
Nottingham
free and open source software for geospatial
Or maybe Finland?
End of Part Two!

- Now you have the tools!