Why depending on the contributions of others while contributing yourself makes sense

Roger Bivand

11 June 2014

1Department of Economics, Norwegian School of Economics, Helleveien 30, N-5045 Bergen, Norway; Roger.Bivand@nhh.no
Overview

1. Introduction
   - Motivation: longer lives

2. OSGeo and R-spatial
   - What does use look like?
   - R-spatial — Building a community
   - R-spatial — sp
   - With a little help . . .

3. Software component stacks
   - Component stacks
   - Dependency challenges
   - Open source geospatial projects

4. Opportunities and challenges
“Why depending on the contributions of others while contributing yourself makes sense” is a long title, but reflects the essence of open source communities.

Of course, nobody is obliged to choose single solutions, and open source software seen as an ecosystem encourages diversity — but beware of overdependence on single components in everything (Mens et al., 2014).

Depending on the contributions of others is rather “passive” in community terms, but often we think that we have nothing important to contribute.

However, “contributing yourself” may be as simple as making your experience available as scripts, forum and mailing list activity, or (when necessary) bug reports.
Using spatial data

- Being explicit about the use of (spatial) data (for example in R) permits attention to be given to the important steps of data handling and representation — rushing to analysis often leads to problems.

- Handling spatial data is never “easy”, because representational choices lead to support consequences, which qualify or undermine inferential outcomes.

- Being close to the data and close to the code may give considerable insight and freedom, and permits co-working with disciplines sharing frames of understanding that are mutually comprehensible.

- Just publishing is not going to be enough; we will need to be able to demonstrate how our conclusions have been reached by releasing code and data (some data is public already).
A news item a year ago described work on overall premature mortality (<75) in England, linking to Public Health England (PHE) sites:
Visualizing longer lives II: forensic reproducible research?

The website does not explain what statistical methods have been used, other than saying that “directly standardised rates” were used. In the absence of contact details, I posted a question under the Connect tab, and received a reply (antedated) some time later:

Roger Bivand - 3 months ago
The spreadsheet data include a Value column, which you map. But they also include Lower_CI and Upper_CI values showing the district-wise spread. Could you please provide a clear link to the description of the statistical techniques used here - is this a Poisson-Gamma model or what? The Denominator column is presumably the population adjusted for age, or is it just population. It would be extremely helpful to be able to see how you got to the Value column from the data provided.

Paul Fryers, PHE - Roger Bivand - 3 months ago
The detailed metadata for the indicators are available on the Public Health Outcomes Framework Data Tool (http://www.phoutcomes.info). The methodology used for the confidence intervals is described there (specifically http://www.phoutcomes.info/pub... with further references for more details.

Roger Bivand OSGeo mutual dependency
Before contacting Paul Fryers by email and meeting him at a spatial epidemiology meeting, it was not easy to grasp how the rates had been constructed, and especially how the class intervals for the maps had been constructed. The data are provided in spreadsheet form, and can be merged with area boundaries (Contains Ordnance Survey data © Crown copyright and database right 2013):

```r
> library(rgdal)
> sm <- readOGR(".", "Prem_mort_sim")
> sm1 <- sm[!is.na(sm$Value),
+     ]
> names(sm1)[16:26]

[1] "Indictr" "Tim_Prd" "Area_Cd"
[4] "Area_Nm" "Value"   "Lowr_CI"
[7] "Uppr_CI" "Count"   "Denmntr"
[10] "Sex"    "Age"
```
If we compare the different classes of local authorities represented in the data set for premature mortality rates per 100,000, 2009-2011, we see considerable variability between class, reflecting some of the structuring of the online maps. Covariates are also available for download, but are not used here.
Using the data provided, and credit should be given to PHE for making it available, we can see that the gap between the upper and lower (95%) confidence intervals is large for observations with small populations and vice-versa. The Denmntr variable is three-years’ worth of the age standardised European Standard Population, it turns out, and the rates (Value) are annual per 100,000:
In conversation, Paul Fryers referred to work (and spreadsheets with macros) on the Association of Public Health Observatories’ website. These appear to have connections to the use of funnel plots to reflect the sample size effect on the rates from smaller aggregate areas. Using formulae from Dover and Schopflocher (2011, p. 3), we can reconstruct the confidence intervals:

```R
> alpha95 <- qnorm((1 - 0.95)/2, +   lower.tail = FALSE)
> se <- sqrt((sm1$Value * (1e+05 - +   sm1$Value))/sm1$Denmntr)
```
Given this insight, we can construct a funnel plot and fumble towards the class intervals used for the published maps of health outcomes:

```r
> alpha998 <- qnorm((1 - 0.998)/2, +     lower.tail = FALSE)
> phat <- 268
> o <- order(sm1$Denmntr)
> se <- sqrt((phat * (1e+05 - +     phat))/sm1$Denmntr)
> sm1$value <- (sm1$value - phat)/se
```
So finally the five class 99.8% and 95% map, and the (politically chosen) four class 95% map split on zero:
OSGeo and R-spatial

1. Introduction
   - Motivation: longer lives

2. OSGeo and R-spatial
   - What does use look like?
   - R-spatial — Building a community
   - R-spatial — sp
   - With a little help . . .

3. Software component stacks
   - Component stacks
   - Dependency challenges
   - Open source geospatial projects

4. Opportunities and challenges
OSGeo and R-spatial

- Before describing how R functionality for using spatial data, let us look at some examples of “autonomous” uses
- Once software is released, it may start “living its own life,” as users express their own needs — in a scripting environment like R, users are offered great freedom
- Following these examples, we will move forward by showing how the R-spatial community and software projects have grown
- This in turn will point to the crucial dependence of R-spatial on the wider OSGeo communities, their projects, software, and willingness to help
What does use of R-spatial look like?

The point of working on classes for spatial data in R is to let users and developers work without having to re-think representation. Some examples:

- The FAO yearbook and their input to the `classInt` package for choropleth map class intervals
- `plotKML` to permit spatial and spatio-temporal results to be displayed in Google Earth
- Using maps in disaster and emergency settings: showing Hurricane Sandy damage
- James Cheshire’s enthusiastic blog and courses in an active community largely in England, with extensions to `ggplot2`
- Oscar Perpiñán Lamigueiro’s considered work in displaying data (and a new book) driven by the needs of work on solar power (and much more)
The 2013 FAO Statistical Yearbook

“The 2013 FAO Statistical Yearbook ... has been created from beginning to end with the statistical software R and the typesetting language LaTeX: from data retrieval, to data processing, indicator construction, and blueprint-ready pdf file for distribution.”
Partly to meet the needs of Global Soil Information Facilities, Tomislav Hengl has been coordinating activity on displaying output on Google Earth (admittedly not open source) in the `plotKML` package ([http://gsif.isric.org/doku.php?id=wiki:tutorial_plotkml](http://gsif.isric.org/doku.php?id=wiki:tutorial_plotkml)). Other packages use R graphical devices with contextual backgrounds from GE and OSM.
In a talk at useR! 2013, Charles DiMaggio presented a talk on using maps in disaster and emergency settings; he had with his students used maps made with R in deploying volunteers in the aftermath of Sandy. The talk is at http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/resources/sandy/useRslides.pdf.
http://spatial.ly/ is a successful and upbeat blog, evangelical in tone: “The software has become established as one of the best around for statistics and it is becoming increasingly recognised as a tool for data visualisation and spatial analysis.”

(http://spatial.ly/2013/04/analysis-visualisation-spatial-data/)
Ease of access and representativeness I

James Cheshire: “Thanks to the release of log files containing all hits to http://cran.rstudio.com/ server it is possible to make a map showing the parts of the world with the most active R users (specifically those mostly using the RStudio interface).”
(http://spatial.ly/2013/06/r_activity/)
Ease of access and representativeness II

Oscar Perpiñán: “More or less explicitly they use the RStudio logs to give an answer to the question ‘How many people use this R package?’ In my opinion, such question cannot be answered safely with the RStudio download logs.
The RStudio mirror is a sample that cannot be safely regarded as representative of the mirrors network. The mirrors of the Comprehensive R Archive Network, mostly operated by public or nonprofit institutions, provide faster package downloads for users at their geographical location” (http://procomun.wordpress.com/2013/06/15/rstudiologs/).
Displaying spatial data

Oscar Perpiñán: “Some time ago . . . I wondered how a multivariate choropleth map could be produced with R. Here is the code I have arranged to show the results of the last Spanish general elections in a similar fashion” (http://procomun.wordpress.com/2012/02/18/maps_with_r_1/).
Bergens Tidende interactive graphics

R can also be used to prepare data for Javascript (including d3 http://d3js.org/), including maps. This is a recent example from my local newspaper concerning immigrant populations in Norwegian municipalities (http://multimedia.bt.no/fjordkloden):
Bergens Tidende interactive graphics

However, when the reporting units are very detailed, it is possible that the unreflective or untrained observer may draw unwarranted conclusions (I produced the boundaries which are Statistics Norway units clipped against the coastline):
R spatial

- Albrecht Gebhardt did a lot of the early porting from S to R; there were also valuable contributions from spatial statisticians at Lancaster University, at UCAR, and many others.

- Kurt Hornik, who runs CRAN, encouraged me to talk about R and GIS at the March 2001 Distributed Statistical Computing meeting in Vienna, at which I got to know active developers personally.

- At a meeting in Santa Barbara in Spring 2002, I met other researchers who provided critical, occasionally very critical, comments and encouragement to continue fostering an R spatial community.

- By the next DSC meeting in March 2003, I was organising a thematic session on spatial statistics, and a crucial fringe developers’ workshop to discuss how to advance spatial data analysis in R.
Since 2003, a number of community-building steps have been made over and above developing contributed packages. From the CRAN side, the Spatial task view is the hub, to which traffic is channelled to package pages and to ancilliary websites, as well as the special interest group mailing list. There is also a task view for handling and analyzing spatio-temporal data.
R-sig-geo mailing list

Following the 2003 workshop, we started a project on Sourceforge to permit joint development, and a mailing list served within the family of R lists from Zurich. Traffic on the list has grown steadily, with a subscribed membership in May 2014 of over 3100. Naturally, many of these “lurk” without posting, while others post without helping, and many fewer help by answering posted questions. This final group is however growing, and since the list archives are also kept on Nabble, they can be searched for information.
R Forge is used actively by individuals and groups in developing packages for spatial data analysis, with 100 projects registered in August 2012. Some projects are registered in more than one topical area, some may never mature, but some are already in active use; the `raster` package is already frequently discussed on R-sig-geo — it was released to CRAN in late March 2010 after a gestation of 16 months.
The **sp** package

In 2003, we agreed that a shared system of new-style classes to contain spatial data would permit many-to-one and on-to-many conversion of representations, avoiding the then prevalent many-to-many conversion problem. The idea was to make it easier for GIS people and stats people to work together by creating objects that “looked” familiar to both groups, although the groups differ a lot in how they “see” data objects. Package dependencies have grown, here the upper diagram (from our book) shows packages depending on **sp** in April 2008, the lower diagram in May 2014 (see also [http://dirk.eddelbuettel.com/blog/2012/08/16#counting_cran_depends_followup](http://dirk.eddelbuettel.com/blog/2012/08/16#counting_cran_depends_followup)).
Spatial objects

- The foundation object is the Spatial class, with just two slots (new-style class objects have pre-defined components called slots)
- The first is a bounding box, and is mostly used for setting up plots
- The second is a CRS class object defining the coordinate reference system, and may be set to CRS(as.character(NA)), its default value.
- Operations on Spatial* objects should update or copy these values to the new Spatial* objects being created
Coordinate reference systems

- Coordinate reference systems (CRS) are at the heart of geodetics and cartography: how to represent a bumpy ellipsoid on the plane.

- We can speak of geographical CRS expressed in degrees and associated with an ellipse, a prime meridian and a datum, and projected CRS expressed in a measure of length, and a chosen position on the earth, as well as the underlying ellipse, prime meridian and datum.

- Most countries have multiple CRS, and where they meet there is usually a big mess — this led to the collection by the European Petroleum Survey Group (EPSG, now Oil & Gas Producers (OGP) Surveying & Positioning Committee) of a geodetic parameter dataset.
Coordinate reference systems

- The EPSG list among other sources is used in the workhorse PROJ.4 library, which as implemented by Frank Warmerdam, handles transformation of spatial positions between different CRS.
- This library is interfaced with R in the `rgdal` package, and the CRS class is defined partly in `sp`, partly in `rgdal`.
- A CRS object is defined as a character NA string or a valid PROJ.4 CRS definition.
- The validity of the definition can only be checked if `rgdal` is loaded.
Spatial*DataFrames are constructed as Janus-like objects, looking to mapping and GIS people as “their” kind of object, as a collection of geometric features, with data associated with each feature. But to data analysts, the object “is” a data frame, because it behaves like one; a data frame is a rectangular data structure with rows of observations on columns of variables, and is very widely used in analysis and visualisation in R.
This table summarises the classes provided by `sp`, and shows how they build up to the objects of most practical use, the Spatial*DataFrame family objects:

<table>
<thead>
<tr>
<th>data type</th>
<th>class</th>
<th>attributes</th>
<th>extends</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>SpatialPoints</td>
<td>none</td>
<td>Spatial</td>
</tr>
<tr>
<td>points</td>
<td>SpatialPointsDataFrame</td>
<td>data.frame</td>
<td>SpatialPoints</td>
</tr>
<tr>
<td>pixels</td>
<td>SpatialPixels</td>
<td>none</td>
<td>SpatialPoints</td>
</tr>
<tr>
<td>pixels</td>
<td>SpatialPixelsDataFrame</td>
<td>data.frame</td>
<td>SpatialPixels</td>
</tr>
<tr>
<td>full grid</td>
<td>SpatialGrid</td>
<td>none</td>
<td>SpatialPixels</td>
</tr>
<tr>
<td>full grid</td>
<td>SpatialGridDataFrame</td>
<td>data.frame</td>
<td>SpatialGrid</td>
</tr>
<tr>
<td>line</td>
<td>Line</td>
<td>none</td>
<td>Line list</td>
</tr>
<tr>
<td>lines</td>
<td>Lines</td>
<td>none</td>
<td>Line list</td>
</tr>
<tr>
<td>lines</td>
<td>SpatialLines</td>
<td>none</td>
<td>Spatial, Lines list</td>
</tr>
<tr>
<td>lines</td>
<td>SpatialLinesDataFrame</td>
<td>data.frame</td>
<td>SpatialLines</td>
</tr>
<tr>
<td>polygon</td>
<td>Polygon</td>
<td>none</td>
<td>Line</td>
</tr>
<tr>
<td>polygons</td>
<td>Polygons</td>
<td>none</td>
<td>Polygon list</td>
</tr>
<tr>
<td>polygons</td>
<td>SpatialPolygons</td>
<td>none</td>
<td>Spatial, Polygons list</td>
</tr>
<tr>
<td>polygons</td>
<td>SpatialPolygonsDataFrame</td>
<td>data.frame</td>
<td>SpatialPolygons</td>
</tr>
</tbody>
</table>
This table summarises the methods provided by `sp`:

<table>
<thead>
<tr>
<th>method</th>
<th>what it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>[</td>
<td>select spatial items (points, lines, polygons, or rows/cols from a grid) and/or attributes variables</td>
</tr>
<tr>
<td>$, $&lt;-, [, ]&lt;-</td>
<td>retrieve, set or add attribute table columns</td>
</tr>
<tr>
<td>spsample</td>
<td>sample points from a set of polygons, on a set of lines or from a gridded area</td>
</tr>
<tr>
<td>bbox</td>
<td>get the bounding box</td>
</tr>
<tr>
<td>proj4string</td>
<td>get or set the projection (coordinate reference system)</td>
</tr>
<tr>
<td>coordinates</td>
<td>set or retrieve coordinates</td>
</tr>
<tr>
<td>coerce</td>
<td>convert from one class to another</td>
</tr>
<tr>
<td>over</td>
<td>combine two different spatial objects</td>
</tr>
</tbody>
</table>
Using **Spatial family objects**

- Very often, the user never has to manipulate Spatial family objects directly, as we have been doing here, because methods to create them from external data are also provided.

- Because the Spatial*DataFrame family objects behave in most cases like data frames, most of what we do with standard data frames just works — like [ or $ (but no `merge`, etc., yet).

- These objects are very similar to typical representations of the same kinds of objects in geographical information systems, so they do not suit spatial data that is not geographical (like medical imaging) as such.

- They provide a standard base for analysis packages on the one hand, and import and export of data on the other, as well as shared methods, like those for visualisation.
Extension of \texttt{Spatial} family objects

Many packages — as we saw a moment ago — depend on \texttt{sp}, fewer extend the classes used there; among them are:

- \texttt{spacetime} provides classes and methods for spatio-temporal data, including space-time regular lattices, sparse lattices, irregular data, and trajectories.

- \texttt{raster} is built around a number of S4 classes of which the RasterLayer, RasterBrick, and RasterStack classes are the most important; a notable feature of the \texttt{raster} package is that it can work with raster datasets that are stored on disk and are too large to be loaded into memory.
The reliance of `sp` classes on the PROJ.4 string representation of coordinate reference systems was noted earlier.

XML-based alternatives could have been chosen, but PROJ.4 strings are also “human-readable”, and readily converted into other representations.

The version of the PROJ.4 library to which `rgdal` is linked is reported to make reproducible research somewhat easier; its behaviour needs careful tracking in a long-running application like R.

In the forthcoming version of the PROJ.4 library, it will be easier to check for metadata files, but it is still very difficult to access their versions, as we have seen recently.
GDAL/OGR

- Although it is possible to import and export geographical data format by format, using GDAL and OGR is very much easier.
- The first raster versions of the `rgdal` package by Tim Keitt were made available in early 2003; it provided bindings to the GDAL geospatial library for reading, writing, and handling raster data.
- Since then, it has been merged with work on coordinate reference system projection and OGR vector reading by Barry Rowlingson, extended to write OGR vector files, and supplied with wrapper functions using `sp` classes to contain the data being imported and exported.
- Because `rgdal` loads GDAL into a long-running application, R, the GDAL error handler is now set to the R error handler immediately before each call to a GDAL function.
GEOS

- Development of the GEOS library interface to R began in late 2009, and made much progress in the 2010 Google Summer of Coding, with Colin Rundel making a large contribution.
- The rgeos package was released on CRAN in March 2011, and is beginning to be used in other packages (34 CRAN packages May 2014).
- One issue uncovered by Colin Rundel in his work on the interface was the importance of the coordinate precision model; integer geometry done with floating point coordinates.
- A specific issue raised in interfacing GEOS (and OGR) is that use is made of the OGC SFS geometry specification, but the SpatialPolygons class in sp is more like a shapefile, without clear assignment of interior rings to exterior rings.
GRASS 6 was released in March 2005, and has now reached 6.4.3; the re-implemented interface package spgrass6 works with GRASS 6 and 7.

spgrass6 is loose-coupled, using GDAL on both sides of the interface to exchange vector and raster data by writing to and reading from a temporary directory.

From April 2009, spgrass6 was revised to support a second mode of operation; the earlier way of using R within a GRASS session was supplemented by the ability to initiate a GRASS session from R, setting up the environment variables used by GRASS.

This was complemented by interfacing most GRASS commands directly in a cross-platform fashion, using the -interface-description flag that GRASS commands use to return their flags, parameters, etc.
Open source geospatial software component stacks

1. Introduction
   - Motivation: longer lives

2. OSGeo and R-spatial
   - What does use look like?
   - R-spatial — Building a community
   - R-spatial — sp
   - With a little help . . .

3. Software component stacks
   - Component stacks
   - Dependency challenges
   - Open source geospatial projects

4. Opportunities and challenges
Software component stacks

Before discussing software component stacks for GC, we should acknowledge the importance of open standards for geospatial data interchange.

Kralidis (2008) points out the importance of concepts such as that of spatial data infrastructure, whether established within national jurisdictions, within supranational jurisdictions, or by international standards organisations.

A fuller coverage of the relationships between spatial data infrastructures and free and open source geospatial software is given by Steiniger and Hunter (2012).

Kralidis (2008) also helpfully distinguishes between formal, de facto, and ad hoc standards, which provide the flexibility needed to move ahead somewhat faster than standards committees are usually able to do.
Component stacks

- Software components appear to have been defined first by McIlroy (1969), as interchangeable subassemblies by analogy with mass production manufacturing.
- The software component stack has been a core concept of programming at least since the publication of Kernighan and Plauger (1976), systematising the experience of Bell Labs computer scientists.
- Some of the lessons are made clear in programming itself (Kernighan and Pike, 1999), while others affect how one may “glue” small utility functions together in an interactive and/or scripting language (Kernighan and Pike, 1984).
- Consequently, a software component stack can be taken as a sequence of component programs that are used together to achieve a common goal.
Component stacks

- Using stacks of components becomes attractive when task objectives can more easily be met by using components developed by others than by developing them independently.

- When the costs of keeping a stack working exceed those of rewriting, the stack may fail, but this is seldom the case.

- Open source software developers often advertise application programming interfaces (API), with an implicit promise that other downstream developers using the API will be less subject to incompatible changes.

- It is then vital that changes in these underlying components do not change the way that dependent components function, unless their earlier behaviour had been in error.
As already noted, developers wishing to integrate software components in stacks must pay careful attention to the versioning of the components, and to the impacts of upstream changes on downstream components.

The terms upstream and downstream refer to the ordering of the components, with data flowing from upstream to downstream components.

If the specification of an upstream component changes, those following it will need to be modified.

If the changes are forced by real bugs being fixed, or security holes being blocked, downstream components must react in appropriate ways.
However, some changes occur for other reasons, such as code cleaning, reimplemention, or the resolution of licence issues in otherwise functioning code.

In most cases, upstream developers then attempt to reduce changes in their interfaces with downstream components to an unavoidable minimum.

Open source projects are typically most constrained with respect to developer time for maintenance, including the revision of functioning code to accommodate upstream changes that may not improve downstream performance.

A particularly troublesome issue for dynamically linked software components in relatively long-running applications is that of thread safety.
Open source geospatial projects

- The Open Source Geospatial Foundation was brought into being in 2006 as a successor to the Mapserver Foundation, itself created the year before.
- In addition to providing a shared infrastructure and procedural framework for web mapping, desktop application and geospatial library projects, OSGeo aims to promote open source geospatial software use and development, including use integrated with proprietary software.
- McIhagga (2008) discusses some of the ways in which communities of practice have developed, with particular reference to web mapping, and in his description, the open source web mapping “ecology”.
- The Geospatial Data Abstraction Library is a crucial part of the upstream geospatial library infrastructure.
Open source geospatial projects

- One of the most important components required by geospatial applications is the provision of robust and clear representations of coordinate reference systems.

- Because GEOS uses OGC SFS specifications for geometries, it does not “build” topologies in the classical GIS arc-node understanding.

- GRASS (Geographic Resources Analysis Support System) was already twenty years old when the GRASS developers collaborated in founding OSGeo, and they have been playing an important role in the broader OSGeo movement (Neteler et al., 2008); many of the more recent developments in GRASS are covered by Neteler et al. (2012).
Opportunities and challenges

1. Introduction
   - Motivation: longer lives

2. OSGeo and R-spatial
   - What does use look like?
   - R-spatial — Building a community
   - R-spatial — sp
   - With a little help . . .

3. Software component stacks
   - Component stacks
   - Dependency challenges
   - Open source geospatial projects

4. Opportunities and challenges
Opportunities and challenges

- The availability of spatial data, and the pervasiveness of geospatial applications, not least in location services, are increasing.
- Existing software components are being used to support these advances, and are being challenged to adapt to new platforms, especially mobile platforms.
- The push towards representing, querying, and analysing spatio-temporal data is of particular importance, given the publication of Cressie and Wikle (2011).
- It will be of great interest to see how the broader TerraLib community develops in coming years, also in its interactions with other open source geospatial communities.
R-spatial: where from here?

- Spatio-temporal classes are at present combinations of (2D) spatial and temporal representations, better than *ad hoc*, but needing development, also in GIS
- Big spatial data probably should not be held in the R workspace, but rather held elsewhere and accessed only when needed (with reference pointers and caching)
- Revisiting spatial classes and methods is needed in the light of these two opportunities for improvement, but maintaining backward compatibility
- We are planning to discuss options at Geostat in mid-June, in particular to see how what has been learnt from the *raster* package can be integrated in development prototypes
Experience and progress

- Experience from R spatial has shown that the nurturing of communities of interest and intention is of fundamental importance.
- One unresolved issue concerns channels for information exchange, in which the aging mailing list technology is straining to keep afloat as newer users prefer hosted fora to search for answers to what they understand to be their questions.
- One challenge is to attempt to sustain community memory, to try to avoid too many repeat solutions being offered to questions that have been resolved.
- The growing importance of open source software is also a return to one of the ways in which research was done when spatial analysis and spatial statistics were first established.
References


