NHH

INSTITUTT FOR SAMFUNNSØKONOMI

DEPARTMENT OF ECONOMICS

SAM 23 2011

ISSN: 0804-6824

November 2011

Discussion paper

Geocomputation and open source software: components and software stacks

BY

Roger S. Bivand

This series consists of papers with limited circulation, intended to stimulate discussion.

Norges
Handelshøyskole

NORWEGIAN SCHOOL OF ECONOMICS

Geocomputation and open source software: components and software stacks

Roger S. Bivand*

November 26, 2011

Abstract

Geocomputation, with its necessary focus on software development and methods innovation, has enjoyed a close relationship with free and open source software communities. These extend from communities providing the numerical infrastructure for computation, such as BLAS (Basic Linear Algebra Subprograms), through language communities around Python, Java and others, to communities supporting spatial data handling, especially the projects of the Open Source Geospatial Foundation. This chapter surveys the stack of software components available for geocomputation from these sources, looking in most detail at the R language and environment, and how OSGeo projects have been interfaced with it. In addition, attention will be paid to open development models and community participation in software development. Since free and open source geospatial software has also achieved a successively greater presence in proprietary software as computational platforms evolve, the chapter will close with some indications of future trends in software component stacks, using Terralib as an example.

1 Introduction

In much the same way that Bivand and Lucas (2000) — a chapter in the first edition of this collection on the integration of models and geographical information systems — was a review of literature, this chapter will consider relationships between geocomputation and open source software. Some of the insights from our earlier work in fact fed directly into the development of interfaces between the open source GRASS GIS and the R statistical language and environment, as initially

*Email: Roger.Bivand@nhh.no, Department of Economics, NHH Norwegian School of Economics, Helleveien 30, N-5045 Bergen, Norway

described by Bivand and Neteler (2000). The structuring of relationships between software components — with ensuing workflow challenges and opportunities — has matured over time, informing geocomputation communities using either open source or proprietary software, or both together.

An aspect of the progress made in software development communities has been the ratio of signal to noise in information diffusion. Books such as Mitchell (2005), Erle et al. (2005) and Gibson and Erle (2006) gave rich insight into myriad possibilities for committed customisers and consultants, but at a distance from what might be termed "mainstream" GIScience; perhaps "hacking" and GIScience are more comfortable at a distance? Applied research often, however, lives between these two places, and needs to find practical solutions to real problems within the constraints of available hardware, software, and programming and scripting competence. It is perhaps a paradox that very little software used to tackle real scientific problems is written by programmers with a background in computer science nowadays; much is written by domain scientists with deadlines to meet.

As many, including recently Rey (2009), have pointed out, the involvement of domain scientists in coding has effectively "included" the code in their research output, making its openness for scrutiny important for the verification of project results and methodologies. Different disciplines approach this question in different ways, with some journals still unwilling to allow software to be cited in references, and unhappy about fully documented software footnotes; others require the submission of supplementary materials including code for the convenience of referees and readers. Access to code to permit research to be reproduced is becoming important in many disciplines, as Leisch and Rossini (2003) show with respect to statistics.

Voices of free and open source software insiders like Ramsey (2007) are important, because they suggest the apparent level of reflection available to those developers closest to the bug-trackers. More reflection is perhaps shown in contributions such as Câmara et al. (2010), but in Ramsey (2007), we are reading a narrative written by a developer with commit rights to major open source geospatial software projects. His distinction between the 'C', the 'Java', and the '.Net' tribes seems well taken, fairly reflecting the ways in which developer communities have evolved; we will return to these communities later in the chapter.

The field of geospatial open source software projects was surveyed in detail by its participants in Hall and Leahy (2008b), and their descriptions constitute a clear picture of the ways in which they see their contributions. Some of the chapters have no references, and are obviously statements by developers with practical rather than academic goals. Other chapters are more similar in character to two other books published in the same year, Neteler and Mitasova (2008) and Bivand et al. (2008), both of which aim to provide applied researchers with guides to the software tools they may find useful in carrying out their work.

This practical approach to the conduct of research is noted by Sui and DeLyser (2011) in the context of academic geography, which one might hope will make helpful contributions in the future after a period of discriminating against quantitative methods even where they were appropriate. Recent years have seen surveys of the potential of open source geospatial software in areas as diverse as health geographics and spatial epidemiology (Fisher and Myers, 2011; Vanmeulebrouk et al., 2008; Yi et al., 2008), landscape ecology (Steiniger and Hay, 2009), water resources management (Chen et al., 2010), and courseware for GIS education (Schweik et al., 2009). Roberts et al. (2010) provide much insight into the ways in which open source and proprietary software solutions intermesh in ecological geoprocessing. Finally, a further general survey is provided by Steiniger and Bocher (2009), in which the categories of the different software varieties, and the range of open source licence conditions are discussed in detail. Here we will accept their broad definition of free and open source software, termed open source for brevity, without further discussion, as the distinctions seem clear; they are also largely shared by Rey (2009), and so do not require repeating at length.

Our task here is rather to review central issues and projects of importance for geocomputation related to open source software, and the enriching of workflows that may be achieved by adding open source components to otherwise proprietary approaches. The open source components are distinguished by the availability of source code under free and/or open source software licences, by access to infrastructures such as version control systems for source code, bug trackers, mailing lists and at least partly organised communities, and by the documentation of external dependencies in the build and install system. As will be shown below, these qualities may vary a good deal across projects, with consequences for the ease of software stacking (or otherwise) experienced in practice.

We will proceed by examining software component stacks for geocomputation first, looking at language environments, component stacks, and crucially at dependency challenges. Next we describe selected open source geospatial projects within the narrow defintion of projects associated with the Open Source Geospatial Foundation (OSGeo), which provides key shared infrastructure for projects, as well as major annual international conferences. Drawing on my own experience, we go on to see how OSGeo projects have been interfaced with the R statistical language and environment, providing examples of how geocomputation may be advanced by using R for programming, scripting and analysis. Alternatively, the Python language and environment, or other candidates, could have been chosen, but my subjective preference is for R. We round off by discussing future prospects.

2 Software component stacks for geocomputation

Before discussing software component stacks for geocomputation, we should acknowledge the importance of open standards for geospatial data interchange. Unless data formats and protocols are agreed, it is very difficult to generate the synergies required for constructive collaboration. 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. The work of the Open Geospatial Consortium (OGC), with members drawn from software companies, research institutes and the broader user community, has been central in this respect. The availability of publically adopted OGC standards has made it possible for software developers of all varieties to share key specifications that enable data to be passed from component to component in controlled ways.

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. The adoption of Keyhole Markup Language (KML) as an OGC standard, based as is Geography Markup Language (GML) on XML, was a wise step, in that it permitted the incorporation of a widely adopted lightweight data representation within a family of standards. Software development benefits from disciplined standards and from rapid but occasionally chaotic progress; we very often need both approaches, and benefit from drawing them together where feasible.

While OGC pays considerable attention to interchange standards, other open standards specifications are of relevance for geocomputation. Dunfey et al. (2006) present an open architecture vector GIS using scalable vector graphics (SVG), easing visualization because of the adoption of this standard by the WWW Consortium. SVG viewers of various kinds have been developed, some closed, some open source, but all capable of rendering the same input data because the specification itself is an open standard. Open source software components may be used in connection with development, but often to “glue” together data in known, sometimes standard, specifications; prototyping using interpreted languages is often a chosen solution. Batcheller and Reitsma (2010) show how open source components may be integrated to permit spatial data discovery through feature level semantics in this context.

While the availability of open standards, and of open source software components, provides us with a great deal of flexibility in application implementation, Schweik et al. (2009) point to advantages in course design and training. The use of open source software for training allows the trainer to tailor the software to the needs of the course, and removes the burden of acquiring and administering software licences. When using proprietary software, in addition to practical costs, the structure of the course is “tailored” by the chosen software, perhaps diverting attention from the core focus.

However, much open source software, in particular desktop GIS, appears to imitate popular proprietary software, for example Quantum GIS (QGIS) and the former ArcView desktop GIS may well perform very similarly in training. In addition, courses are often obliged to take into account the needs of participants to acquire familiarity with established proprietary systems before starting work where these systems are deployed as standard. The tension between generic GIS and geospatial training giving graduates general skills, and software specific training is very real, especially where the software presupposes the dominance of a graphical user interface. Where generic skills are taught in relation to scripting, interpreted languages, and command line interfaces, the needs of participants to acquire abilities that can be applied at work from day one may be readily met using any suitable mixture of open source and proprietary software.

Steiniger and Bocher (2009) and Chen et al. (2010) give recent overviews of open source GIS software, but with constraints on what they see as general suitabilities and functionalities. It seems that their preference for applications rather than component stacks has affected the ways in which software is perceived. Preferences for graphical user interfaces (GUI) has in particular obscured the fact that developing GUIs absorbs a great deal of developer effort, and that most open source projects face their hardest constraints in mobilising and precisely deploying developer effort. Typically, open source projects face choices between GUI toolboxes, with some developers prefering one cross-platform toolbox, others prefering alternatives. All such projects hit road bumps when the chosen toolbox “upgrades” in a way that is not backwards-compatible, meaning that much GUI work has to be repeated, and possibly supported for both the older and the newer toolbox versions.

In the remainder of this section, we will consider the importance of programming language environments, of component stacks and mechanisms for joining components together, and finally the challenges that arise from trees of dependencies engendered between components.

2.1 Language environments

Câmara et al. (2010) following Ramsey (2007) distinguish between the language environments characterising open source geospatial software. Many projects use the compiled C and/or C++ languages; in the latter case, use varies between projects using modern C++ with templates, and others using C++ more as C. Historically, the adoption of compiled languages by projects has been influenced by the availability of suitable compilers and linkers across the target operating systems and hardware platforms. The emergence of the GNU compiler collection (GCC), and especially the gcc C and the g++ C++ compilers across multiple platforms and operating systems, has made it much easier to ensure that computations made using the same source code do not give platform-dependent output on the same data. This is still not guaranteed, as for example time and time zone handling may differ between operating systems.

The contribution of individuals here is often crucial; the R Windows FAQ 3.1.10 reflects this: "The assistance of Yu Gong at a crucial step in porting R to MinGW-w64 is gratefully acknowledged, as well as help from Kai Tietz, the lead developer of the MinGW-w64 project".[1] Without their important interventions, it would not have been possible to progress with a GCC-based 64-bit R for 64-bit Windows platforms. Not infrequently, such interventions occur unexpectedly, suddenly opening up apparently blocked avenues. Platform-specific open source projects may use compilers supplied with operating systems, some of which are available without charge.

Beyond the compilers and linkers provided with GCC, many projects using C also use the legacy Unix make command to manage the build process, and GNU autoconf to configure the build process by auto-detecting the presence and versions of software dependencies. In addition, many also use GNU libtool to assist in writing input files for make processes on the fly. Others choose a more modern open source build system, CMake; it is however rarely the case that experienced open source developers feel comforable in both build environments. It is much easier for developers to use the same compiler and build train across platforms, so that test suites can be deployed and used in the most convenient way.

Other open source geospatial projects use Java, which handles cross-platform portability by running byte-compiled programs on platform-specific virtual machines. Java was often adopted by projects initiated when the portability of C compilers was in doubt, and where developers felt that a more modern interpreted language was an advantage. JavaScript used as a mechanism for embedding computation in information delivered to web browsers and similar front-end software has become extremely powerful. The initial Google Maps applications programming interface (API) was written as a JavaScript API, but has subsequently been enlarged to include other components.

Before returning briefly to web and mobile geospatial applications, we must note the significance of other major language environments. Ramsey (2007) mentions those based on .Net, with their use of the wrapping of C/C++ and Java components. The Simplified Wrapper and Interface Generator (SWIG) has been developed to permit compiled components to be used in scripting languages such as Perl, Python, PHP, Tcl, and Ruby, among others. Some applications have designed customised interfaces like GRASS with Python; others use calls to the operating system to execute external programs. Shell scripts, known as batch programs on

[1] `http://cran.r-project.org/doc/manuals/R-admin.html#g_t64_002dbit-Windows-builds`.

Windows platforms, have long been a staple form of application integration that have been easy to write and maintain. These are likely to remain of major importance on all platforms; despite appearances, shell scripts are just as convenient on OSX platforms as on other versions of Unix.

2.2 Component stacks

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. They point out that modularization and simplicity in coding lead to greater robustness, because small functions and applications can be tested more thoroughly than large ones. 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). In Bentley et al. (1986), McIlroy shows how Bentley's programming challenge — to tabulate word frequency in a given text — was solved elegantly in a monolithic program by Knuth, but can also be answered using a very short shell script using well-tried small utility programs available in any Unix distribution.

Consequently, a software component stack can be taken as sequence of component programs that are used together to achieve a common goal. The most widely used example is LAMP: Linux, Apache, MySQL and Perl/PHP/Python, comprising a sufficient and capable stack for running a web server with server-side page processing. The languages used here vary, with applications written in C, some C++, and bound together with shell scripts for administration, SQL for data handling, and a scripting language to process web pages dynamically.

As in a jigsaw puzzle, the interfaces between software applications in a stack need to be clear and well-defined. In the LAMP case and similar cases, the interface definitions were both clear and stable, leading to the creation of a critical mass of system administrators, and thus a sufficiently large user base to generate a helpful flow of bug reports. Interfacing applications typically reveals implementation assumptions that are neutral in nature in themselves, but when confronted with unspecified assumptions in interfaced components, may become problematic.

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 fai. This is seldom the case, as reimplementation is fraught with difficulties, especially of ensuring a sufficiently large user base to generate bug reports, and to encourage other developers to join in.

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. This permits them to make the improvements deemed desirable, or necessary bug-fixes, without downstream software being affected. Naturally, software closer to the user interface, or to the web server, will often depend on underlying libraries and services, for example for parsing XML. 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.

Open source software is characterised not only by frequent releases of components, and by rapid bug-fixing leading to patched releases, but also by the availability of checkout from version control systems. This permits developers of "downstream" software to build and test against the current trunk revisions of "upstream" components where necessary, or at least before release, to attempt to future-proof the "downstream" component. Build support systems, such as GNU autoconf, will then set compile flags to indicate the versions of "upstream" components, and/or use will be made of self-declaring version functions to branch on version internally.

Many of these issues have been influenced over recent years by the commissioning of specialist support and customization from open source geospatial developers by customers, or by the opening of existing software codebases. Because large companies often need to integrate multiple software components within specific quality assurance support systems, they contribute code, bug fixes, and contracted development which benefit all users of the components in question. The range of interaction is large, especially because of the rapid growth seen in the use of geographical data.

Web, navigation and mobile geospatial applications have burgeoned in recent years, effectively obscuring most of what the geocomputation community has been concerned with over the past half century. The vast majority of map applications do not include any analysis, and most users of the applications, and associated hardware and software are scarcely aware that their searches, GPS-registered movements, or uses of smart transit passes, constitute data. Mobile devices may be tracked from base stations, but as they also acquire GPS, they can themselves record user positions. Android developers of course can benefit from open source software, and application build trains, but these uses are not strongly connected with most geocomputation. Exceptions include the use of sensor networks and animal tracking, to which we will return below.

Another is the application programming interface in OpenStreetMap (OSM), which supports data input from volunteer contributors, rather than the elaborate visualization and search interfaces provided by leading web, navigation and mobile geospatial applications. Figure 1 shows the OSM component overview, which is not untypical in its complexity. Without the availability of the components developed outside the OSM community, it would have been extremely hard to have achieved the progress we can all see and benefit from in the rapid updating of street maps,

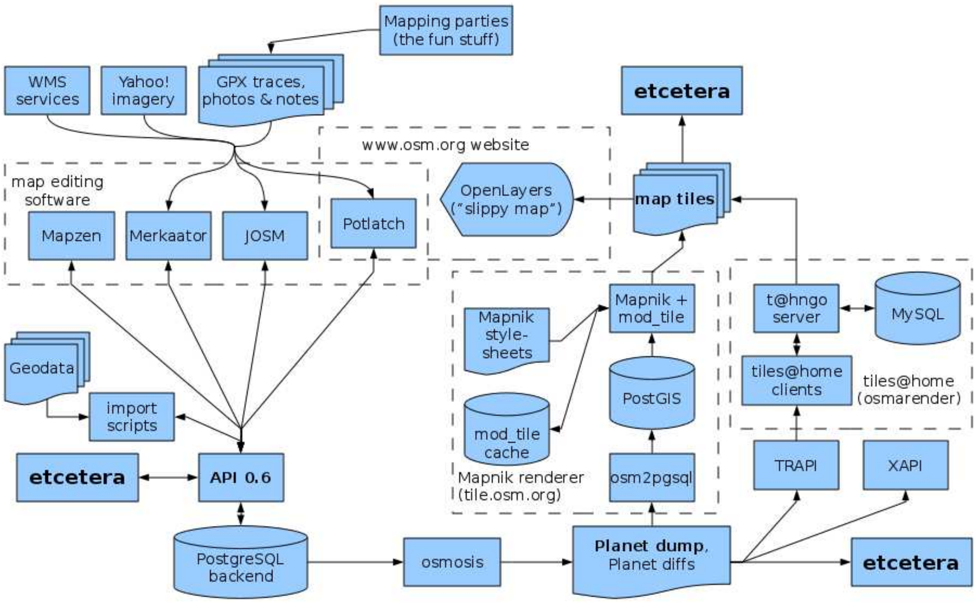

Figure 1: OpenStreetMap component overview, downloaded from `http://wiki.openstreetmap.org/wiki/Component_overview`

especially in places without adequate mapping agencies. It is not coincidental that the 2011 State of the Map conference, focused on OSM, and the 2011 FOSS4G OSGeo conference have been held consecutively in Denver, Colorado.

2.3 Dependency challenges

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. 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, reimplementation, 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. This has been seen often enough when GUI toolkits are chosen — if the toolkit APIs change often, they will be seen as unattractive. The same applies to language and compiler versions; the new versions may be better engineered, but may not be as prevalent on user systems than their predecessors. Python seems to be a case in point, with most Windows geospatial software bundling their own copies, which may complicate updating and maintenance on user systems.

A particularly troublesome issue for dynamically linked software components in relatively long-running applications is that of thread safety. If the upstream component has a global error handler, it may be that multiple downstream components will compete in resetting it to hand off errors to their own error handlers. The same may occur with the setting of global variables. Even if components may be written, or often adapted from earlier code, to be thread safe in themselves, it may be that thread handling in downstream components makes different assumptions. Modern language environments, such as Haskell, attempt to attack this problem at its root, but total reimplementation of complex component stacks is most often not a feasible option.

Defensive use of static linking is a possibility, but places the responsibility for critical updating on the downstream developers in return for control over the dependency in distributed binaries. Alternatively, the downstream component may simply bundle the source code of the upstream components; this is taken to considerable lengths by Boost[2] and its community — Boost provides free peer-reviewed portable C++ source libraries written as collections of header files.

It is often convenient for users to install and maintain binary components rather than to install from source. This then transfers the responsibility for trying to keep component stacks working together to those who package and distribute binary components, such as the `OSGeo4W` project[3] to provide Windows installers and components, or the provision of OSX frameworks[4] for open source geospatial software. There are a number of similar Linux repositories, providing component binary packages, such as DebianGIS[5] and UbuntuGIS,[6] among others. The packagers may also get overenthusiastic and release binaries of early development versions of software, perhaps solving one problem, but leaving others open.

Dependency issues may degenerate into dependency “hell” when downstream necessary components in a stack change so as to have conflicting version dependencies on the same upstream component. If the packaging metadata is not carefully crafted, updating may lead to a component stack failing, or losing stability. Since users often see proposed updates as offering greater security and/or functionality, their presumption will be to update, and trust the metadata to protect them against

[2] `http://www.boost.org`.

[3] `http://osgeo4w.osgeo.org/`.

[4] `http://www.kyngchaos.com/software/frameworks`.

[5] `http://wiki.debian.org/DebianGis`.

[6] `https://wiki.ubuntu.com/UbuntuGIS`.

unanticipated consequences. Writing packaging metadata and binary build systems is another area in which open source projects typically lack developer capacity, because it is both hard and unrewarding. Users take the providers for granted until something gets broken, at which point they complain, understandably reducing developer motivation to offer time to such services.

3 Open source geospatial projects

The Open Source Geospatial Foundation (OSGeo) was brought into being in 2006 as a successor to the Mapserver Foundation, itself created the year before.[7] 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. Its incubation procedure for projects includes legal verification steps to check that code is properly copyrighted and licensed, and that the conditions of use are clear. Many of the geospatial library projects offer code under X/MIT, LGPL, or other licences permitting the distribution of linked builds of closed source downstream components containing modified upstream components.

McIhagga (2008) discusses some of the ways in which communities of practice have developed, with particular reference to web mapping, in his description, the open source web mapping "ecology". Chen and Xie (2008) show how open source SQL data bases with spatial extensions fit into the bigger picture; this is very evident also from Figure 1. There is also a good deal of excitement around the use of non-relational databases with spatial data, such as GeoCouch[8] extending CouchDB; others are also being presented at the OSGeo meeting in 2011.

The PostGIS spatial extensions to PostgreSQL are widely used; PostGIS is licensed under the Gnu General Public License (GPL), while PostgreSQL itself is licensed under its own licence, which is similar to the MIT licence. Software licensed under the GPL is termed Free Software, because licensees are required to make available modified source code if they also publish binary versions of the software for sale or otherwise. Software with more "liberal" licences does not oblige licencees to contribute back to the community if they publish binary software, although many do anyway. The term Open Source software includes Free Software as a strict subset, that is all Free Software is Open Source, but not all Open Source is Free in the understanding of the GPL.

The following review does not attempt to be exhaustive, but rather to establish a basis for the next section, in which links with R will be presented.

[7] http://www.osgeo.org/content/foundation/about.html.

[8] https://github.com/couchbase/geocouch/.

3.1 Geospatial libraries

One of the central geospatial libraies closely associated with geocomputation in its development motivation is GeoTools.[9]. Turton (2008) describes its progress from beginnings in a doctoral research project in Leeds up to about four years ago, and its position as a major upstream component for both desktop applications and web mapping applications written in Java is, if anything, even stronger now. It builds on other components, such as the Java Topology Suite,[10] but implements its own code for spatial reference systems in Java based on the OGP ESPG[11] database. The R **cshapes** package (Weidmann et al., 2011) bundles JTS run through **rJava** for polygon boundary line generalization and distance calculation, but is probably the only R geospatial package using open source geospatial Java components (Weidmann and Gleditsch, 2010).

The Geospatial Data Abstraction Library (GDAL, pronounced GooDAL, with stress on the oo, because it was intended to be object-oriented)[12] is a crucial part of the upstream geospatial library infrastructure. Downstream components needing to read raster data can instead read from the abstracted object representation, rather than being obliged to implement interfaces to each format separately. As Walter et al. (2002) describe its beginnings in relation to the OpenEV desktop application, it simplified reading and writing raster data.

Warmerdam (2008) provides a rounded description of the library, including its OGR vector extensions and design goals. Use is made both of OGC Simple Features specifications, and of the PROJ.4 cartographic projections library. GDAL utilities are provided to give command line access to library functionality; Luis (2007) shows how GDAL and GMT can be combined for exploring grid data. GDAL is also available in interpreted languages like Python and Perl. Its C API is stable, but, as Warmerdam (2008, pp. 99–100) points out, the C++ application binary interface is very dependent on the version of the compiler in particular, termed ABI fragility.

It is not hard to contribute new drivers if the file or web service formats are fully specified, and/or supported by external libraries; I have collaborated in writing a driver for SAGA raster files, and the C++ coding involved was not demanding once the format was documented. GDAL aims to open and read files simply based on regular file characteristics, so that the format used may in fact be transparent for the user. Writing files may be harder, and fewer drivers support file creation and copying than reading. Only a very few XML based vector formats in default builds, such as KML, can be written but not read. Many drivers require the use of external libraries, especially where the external dependency encodes proprietary

[9] `http://www.geotools.org`.

[10] `http://www.vividsolutions.com/jts`.

[11] `http://www.epsg.org/`.

[12] `http://www.gdal.org`.

formats in a closed-source binary shared object, or where it seems wiser not to internalise complete driver code in GDAL itself, only providing stubs linked to library functions.

In conclusion, Warmerdam (2008) mentions the difficult questions of thread safety and internationalization, neither of which have been resolved. The latter issue affects the OGR vector part of the library, as feature attributes are much more likely to use multi-byte characters and/or different codepages. The choice of UTF-8 support is typical of many open source projects, as it falls back to ASCII when only 7 bits convey meaning. Error messages and documentation should also be available in other languages.

The Java Topology Suite has been ported from Java to C++ as GEOS[13] (Geometry Engine — Open Source), including all the OGC Simple Features for SQL spatial predicate functions and spatial operators; like JTS, GEOS assumes planar geometries. GEOS and JTS also share precision models that can be set and retrieved by applications — not infrequently, changing the precision model can affect the results of computation. Because GEOS uses OGC SFS specifications for geometries, it does not "build" topologies in the classical GIS arc-node understanding. The operations are conducted on topologies built on-the-fly and discarded; prepared geometries may be made, speeding operations, and Sort-Tile-Recursive (STR) trees can also be built for querying geometries. It is required that geometries meet SFS specifications. The library is used by PostGIS to provide predicate functions and topology operations, and can be compiled into GDAL to make these operations available for OGR layers. GEOS has been modified to achieve thread safely by the provision of a handle in the C API that is specific to the thread; before long, the thread safe versions will be the only supported functions in the API.

One of the most important components required by geospatial applications is the provision of robust and clear representations of coordinate reference systems. A text representation was introduced in the PROJ.4 library,[14] and pre-dates the OGC well known text (WKT) spatial reference system (SRS). It supports datum transformation in addition to projection, and is part of the OSGeo MetaCRS project encompassing several projection and coordinate system related technologies.[15] Extensive use is made of the OGP EPSG[16] database to encode distinct coordinate reference systems. Extensions to this SRS database, for example used in the ESRI ArcSDE interface, appear to have an uncertain legal status, and do not seem to be available to open source applications in the same way as described in the EPSG Geodetic Parameter Registry terms of use.

[13] `http://geos.osgeo.org`.

[14] `http://trac.osgeo.org/proj`.

[15] `http://trac.osgeo.org/metacrs/`.

[16] `http://www.epsg.org/`.

Chen and Xie (2008) describe the rationale underlying PostGIS[17] as a library of spatial extensions for the PostgreSQL object-relational database system. Because PostGIS uses the OGC Simple Features Specification for SQL, and incorporates the GEOS geometry engine, it makes the underlying database into a powerful spatial data engine and repository, particularly when carefully indexed. PostGIS 2.0 will offer support for raster data, on which development is continuing actively.

TerraLib[18] is positioned as middleware between a chosen object-relational database system and a front-end application. It can store and retrieve spatial data, including raster data since its inception, and apply functions and operations to the data, storing output in the database and passing it to the front-end application for display (Câmara et al., 2008). Its next version, TerraLib 5, will be more tightly integrated with central OSGeo libraries, will support non-DBMS data sources such as web services, and will permit spatio-temporal data to be represented and queried.

3.2 Desktop applications

The best documented open source geospatial desktop application appears to be GRASS GIS (GRASS Development Team, 2011). 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).[19] The GRASS book (Neteler and Mitasova, 2008) is already in its third edition, covering the current GRASS 6 release, which is now at 6.4.1, and has advanced far beyond the book. From its original shell-scripted command line interface form, GRASS now has a legacy open source Tcl/Tk GUI, and a modern wxPython GUI using Python as its scripting language and the wxWidgets open source cross-platform GUI toolkit. In GRASS 7, Python will replace shell scripts for scripting, removing the need to emulate Unix in workflows.

Because of its flexibility, GRASS has been customised for very many different platforms; Sorokine (2007) shows how parallel high-performance visualization may be made available for tiled wall displays. Rocchini et al. (2011) customise GRASS to rectify aerial photographs as a basis for constructing landscape composition indices for tracking climate change. GRASS is used in compute-intensive research in ecological and environmental studies, such as the simulation of the management of alien plants by Roura-Pascual et al. (2009) and Krug et al. (2010). Roiz et al. (6) analyse the factors potentially driving the invasion of tiger mosquitoes in northern Italy under climate change scenarios. Finally, GRASS now has a convenient exten-

[17] `http://www.postgis.org`.

[18] `http://www.terralib.org`.

[19] `http://grass.osgeo.org/`.

sion mechanism, so that additional toolsets can be combined with those distributed with the base system; Jasiewicz and Metz (2011) provide a toolkit for Hortonian analysis of drainage networks. The extension mechanism does not yet support forward-compatibility control checking, so extension authors need to remember to keep their contributions updated.

The Quantum GIS (QGIS)[20] desktop application, like open source Java-based desktop GIS such as gvSIG,[21] uDig[22] and OpenJUMP[23] may appear to the user to resemble proprietary desktop GIS. The GUI structure designs, and in many cases the names given to menu items, seem aimed to ease the path of the novice user moving between open source and proprietary applications. This is also evident in the style chosen by Sherman (2008) in his book on QGIS, fitting a user guide or manual template rather than an academic one. Of course, good academic work is done with these systems, such as Robertson et al. (2009) and Robertson and Farmer (2008), who report on mountain pine beetle infestation in British Columbia, and Cagnacci and Urbano (2008), showcasing a system for handling GPS collar data. QGIS both benefits and suffers from a plugin system, because the plugins add functionality, but may cease working as new versions are released, especially if the plugins rely on external software.

It is worth noting that GRASS and QGIS are OSGeo projects, and both rely on the maintenance and development of the underlying geospatial libraries, such as GDAL and PROJ.4. These dependencies are shared with an important non-OSGeo desktop GIS, SAGA GIS.[24] SAGA has been freshly written in a modular form in C++, and has a command line interface from the shell as well as a GUI. The GUI differs in its use from proprietary GIS, but once differences are noted, is very flexible; it is good at displaying large data sets, and has many analysis modules. Goetz et al. (2011) show how SAGA can be integrated with other tools for modelling landslide susceptibility.

The Integrated Land and Water Information System (ILWIS)[25] is another desktop GIS application, and was released as open source after its commercial existence was terminated in 2005. Other desktop applications include GeoVISTA Studio, described by Gahegan et al. (2008), a problem solving environment; like some others, this application seems mainly intended to support research into GIS and GIS use, especially for visualization and knowledge discovery.

[20] `http://www.qgis.org`.

[21] `http://www.osgeo.org/gvsig`.

[22] `http://udig.refractions.net/`.

[23] `http://www.openjump.org/`.

[24] `http://www.saga-gis.org`.

[25] `http://52north.org/communities/ilwis`.

3.3 Web mapping and services

Lime (2008) describes the evolution of MapServer[26] in some detail. It is an OSGeo project of considerable importance, and its foundation was the entity that was transformed into OSGeo in 2006. It is based on simple principles, but has also been extended with the MapScript scripting language, which may be compiled with the GEOS library to provide considerable server-side analytical power. The MapChat[27] web application (Hall and Leahy, 2008a; Hall et al., 2010) builds on Mapserver; it is an open source tool for integrating maps with real-time and asynchronous discussions between multiple users, who can annotate maps to communicate information. It uses PostGIS as a spatial database backend, and OpenLayers[28] for client-side map display — OpenLayers is a JavaScript library that is also an OSGeo project.

Web mapping services include several other application areas. In addition to web map services (WMS) to serve rasterised data, web feature services (WFS) to serve features (vector data), OGC has defined web processing services (WPS), in which a server accepts geospatial input data, processes it, and transmits the results to the client or another addressee. INTAMAP[29] shows how such a WPS may be configured, here for providing an interoperable framework for real time automatic mapping of critical environmental variables (Pebesma et al., 2011b). Another example of an OGC web standard is for an OGC Sensor Observation Service Client, described by Nüst et al. (2011), and written as an R package **sos4R**.[30]

MapGuide Open Source[31] is another OSGeo project, and like ILWIS, it has its background in a proprietary application. Bray (2008) describes its development as a modern web-based geospatial platform. It uses an OSGeo library not discussed above, Feature Data Objects (FDO), which is an API for manipulating, defining and analyzing geospatial information that is completely data store agnostic. This permits web service providers to be used as a data source, for example WMS and WFS. GeoServer[32] is a Java-based server that permits geospatial data to be viewed and edited. It is also an OSGeo project, and uses OpenLayers and GeoTools. It offers WMS and WFS interfaces, allowing clients to access data.

The WPS and SOS examples both use R as part of their workflow, providing a convienient introduction to the next section, in which we will show how geospatial software components may be used with the R language and environment.

[26] `http://www.mapserver.org/`.

[27] `http://mapchat.ca/`.

[28] `http://openlayers.org/`.

[29] `http://www.intamap.org`.

[30] `http://www.nordholmen.net/sos4r/`.

[31] `http://mapguide.osgeo.org/`.

[32] `http://geoserver.org/`.

4 OSGeo and R-spatial

The R open source programming language and environment (R Development Core Team, 2011) is understandably associated with data analysis and statistical programming. As a general interpreted programming language, it is not limited to these tasks, and can be applied to computational tasks of many kinds. As an example, R can be embedded within the PostgreSQL database as the procedural language PL/R[33]; it can also be interfaced with Python and other languages. In the other direction, C, Fortran, and C++ libraries may be interfaced with R functions through simple mechanisms, which also permit access to R objects in compiled languages, and call-back to R from compiled functions.

The class and method definitions used in R are covered in detail by Chambers (2008), and permit data objects to be represented and handled. These definitions advance in waves, with many fitted model objects using old-style classes, and many data objects using new-style classes. Old-style and new-style generic methods have also recently been brought closer together. A further innovation of some note is the **Rcpp** package, providing a more modern interface between R code and compiled C++ code, described by Eddelbuettel and Francois (2011), and in a Google TechTalk.[34]

Fox (2009) reports on the development of the R project from a sociological viewpoint, based on semi-structured interviews carried out in 2006 and 2007. He points to salient features of the social organisation of the project that have enabled it to provide both a stable platform with strong continuity in functionality, and a rich community encouraged to contribute software packages extending the base functionality.

Prospects and challenges in R package development are discussed by Theußl et al. (2011); they address some of the issues raised above with regard to the consequences of version and platform drift for community contributed add-ons. Contributed packages distributed through the comprehensive R archive network are now very numerous, and have complex dependency structures. Cross-platform checks run on CRAN packages using multiple versions of R from released to development need careful planning to capture inter-package dependencies correctly, and to minimise the need for administrator intervention when things break, which they inevitably do when changes are made.

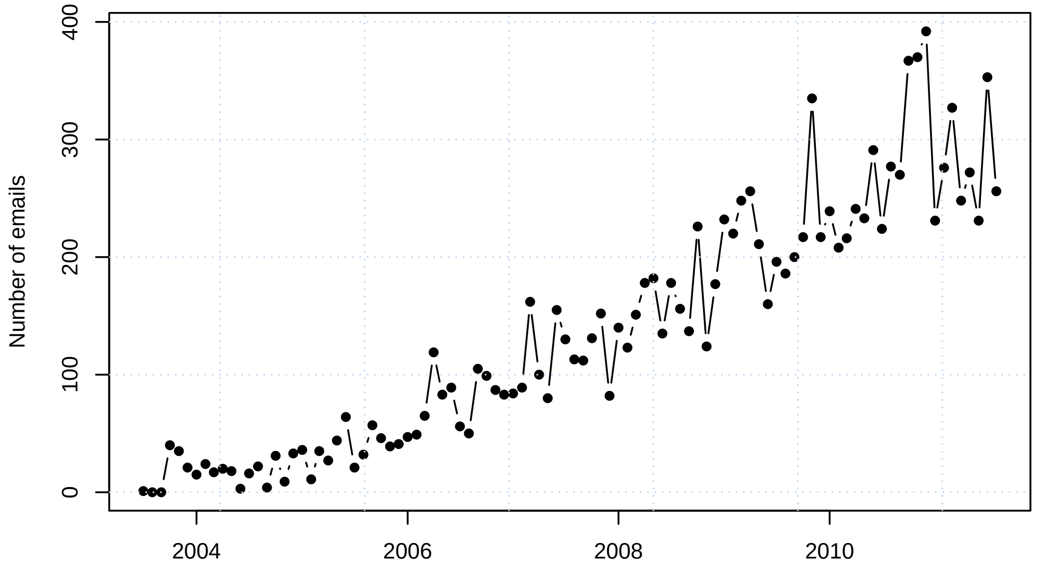

Figure 2: Monthly numbers of emails on the R-sig-geo mailing list, 2003–2011

4.1 R-spatial — sp

In Bivand et al. (2008), we discuss the background for providing spatial data analysis functionality in the R environment, and how the need emerged for classes for spatial data. From 2003, we attempted to make available mechanisms permitting a user and developer community to grow. The R-sig-geo mailing list now has over 2200 subscribers, and Figure 2 shows the steady growth in the numbers of messages exchanged since its inception. It is now the specialised R list with most traffic (excluding the main R-help and R-devel lists). Mailing lists remain a vital part of open source communities, connecting users with each other and developers, encouraging users to become developers, and providing a searchable archives of messages (over 13,000 messages in the case of R-sig-geo).

The **maptools** package (Lewin-Koh et al., 2011) predates the **sp** package, which was released in April 2005, and provided definitions of classes for spatial data (Pebesma et al., 2011a). **maptools** has been adapted to use **sp** classes; it also provides coercion methods between **sp** classes and other spatial data representations in other packages. The intuition underlying the design of **sp** classes has been that

[33] `http://www.joeconway.com/plr/doc/index.html`.

[34] `http://www.youtube.com/watch?v=UZkaZhsOfT4`.

Table 1: The family of **sp** classes.

data type	class	attributes	extends
points	`SpatialPoints`	none	`Spatial`
points	`SpatialPointsDataFrame`	`data.frame`	`SpatialPoints`
pixels	`SpatialPixels`	none	`SpatialPoints`
pixels	`SpatialPixelsDataFrame`	`data.frame`	`SpatialPixels` `SpatialPointsDataFrame`
full grid	`SpatialGrid`	none	`SpatialPixels`
full grid	`SpatialGridDataFrame`	`data.frame`	`SpatialGrid`
line	`Line`	none	
lines	`Lines`	none	`Line` list
lines	`SpatialLines`	none	`Spatial`, `Lines` list
lines	`SpatialLinesDataFrame`	`data.frame`	`SpatialLines`
polygon	`Polygon`	none	`Line`
polygons	`Polygons`	none	`Polygon` list
polygons	`SpatialPolygons`	none	`Spatial`, `Polygons` list
polygons	`SpatialPolygonsDataFrame`	`data.frame`	`SpatialPolygons`

applied statisticians tend to "see" data as represented in rectangular tables, in R `data.frame` objects. Spatial analysts "see" data as rasters or "shapefiles". If these researchers are to work together productively, their perceptions of their data should not be changed but rather accommodated. The **sp** classes behave like `data.frame` objects (when attribute data is present), but can also be handled and visualized as raster or vector data in a "spatial" way. Table 1 shows the range of data objects supported, including points, lines and polygons for vector data, and regular grids for raster data; the `SpatialPixels` representation is a regular grid representation recording cell centre coordinates, but dropping observations with no observed attribute data.

Table 2 lists some of the methods provided in the **sp** package; the most important are undoubtedly the access and assignment functions `$`, `$<-`, `[[`, `[[<-`, which permit the `Spatial*DataFrame` objects to behave as `data.frame` objects. This means that visualising relationships between raster or vector attributes, or fitting a regression model to such attributes, involves no extra steps. One important component of **sp** class objects is the coordinate reference system slot, represented in the form of a PROJ.4 string; it may be set as missing, but is used in both visualization and analysis when specified.

In the figure in Bivand et al. (2008, p. 5) and reproduced on the book website,[35] 23 packages depended on **sp**, of which 8 were written and/or maintained by the book authors. Figure 3 shows that there were 97 CRAN dependencies on **sp** in

[35] `http://www.asdar-book.org/code.php?chapter=0&figure=0`.

Table 2: Methods for **sp** classes.

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

September 2011, and a further 247 suggested **sp** but did not depend on, or import from it. This may be taken as an indication that the provision of classes for spatial data has achieved its goals, to make it easier for researchers to get their own work done without having to reinvent data representations.

In the short time since its publication on CRAN in March 2010 following 16 months in R-Forge,[36] the **raster** package (Hijmans and van Etten, 2011) has been adopted by many users. van Etten and Hijmans (2010) have already published using the package, and undoubtedly many papers will follow. For many purposes, the abstractions introduced in the package simplify the `SpatialGridDataFrame` representation from **sp**, and because only tiles of the raster, raster stack, or raster brick are held in memory, much larger data sets may be handled. The package also provides ways of undertaking map algebra on rasters, including focal operations. It uses the **rgdal** package, among other data feeds, for reading from and writing to files; we turn to this package next.

4.2 Geospatial Data Abstraction Library (GDAL/OGR) and PROJ.4 — rgdal

The first raster versions of the **rgdal** package by Tim Keitt were made available in early 2003, and entered CRAN in late 2003 (Keitt et al., 2011). 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 be-

[36] `https://r-forge.r-project.org/`.

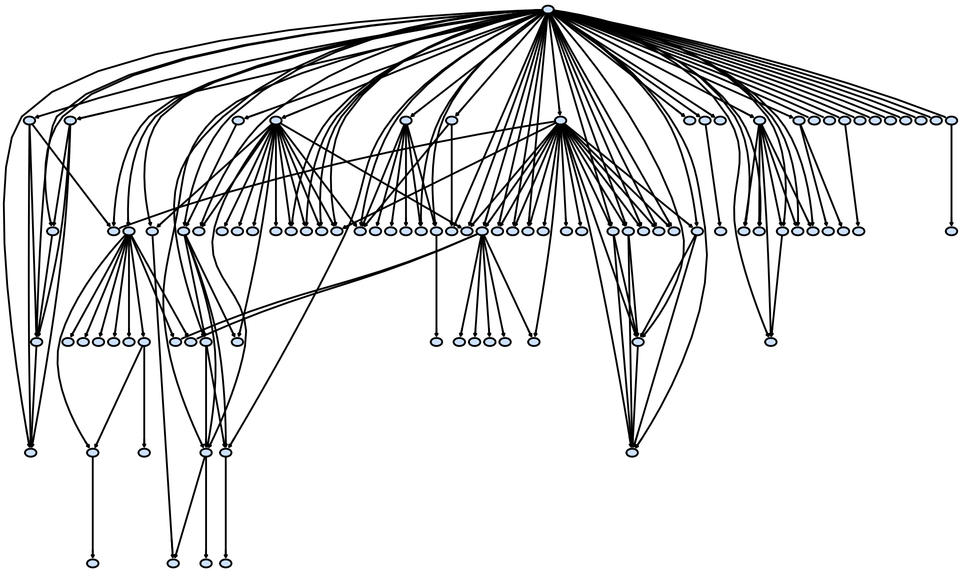

Figure 3: Dependencies on **sp** from CRAN packages, September 2011.

ing imported and exported. Coordinate reference system projection was handled by building against the PROJ.4 library, not least because GDAL itself requires the same library. 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, and restored immediately on function exit to try to ensure thread safety, because of a report of error handler confusion when R and **rgdal** were loaded into QGIS as a Python plugin. When component stacking reaches these levels of complexity, caution is required.

Since R 2.2, a Windows 32-bit binary of **rgdal** has been available from CRAN, thanks to help from Brian Ripley and Uwe Ligges, and since R 2.12 a Windows 64-bit binary version is also available. These are statically linked to GDAL, PROJ.4 and Expat,[37] an open source XML library used for reading KML and GPX files. Brian Ripley has also made an OSX Intel 32+64-bit binary package available on CRAN Extras since R 2.12; OSX binary packages are also available from Kyngchaos[38] thanks to William Kyngesburye. The drivers available in the binary packages are limited to those for which external dependencies were satisfied when the installed images were made, but meet most users' needs for file import and export.

[37] `http://expat.sourceforge.net/`.

[38] `http://www.kyngchaos.com/software/frameworks`.

The use of the package is covered in Bivand et al. (2008, pp. 89–97), and exemplified throughout the code examples from the book. Using the 8 county New York State leukemia data set from Waller and Gotway (2004), we can download and unzip it to a temporary directory, before reading with `readOGR`:

```
> td <- tempdir()
> download.file("http://www.asdar-book.org/datasets/NY_data.zip", destfile=paste(td, "NY_data.zip", sep="/"))
> unzip(paste(td, "NY_data.zip", sep="/"), exdir=td)

> library(rgdal)
> NY <- readOGR(dsn=td, layer="NY8_utm18")

OGR data source with driver: ESRI Shapefile
Source: "/tmp/RtmpQiUQR0", layer: "NY8_utm18"
with 281 features and 17 fields
Feature type: wkbPolygon with 2 dimensions

> class(NY)

[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"

> proj4string(NY)

[1] " +proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs"

> NY_ll <- spTransform(NY, CRS("+init=epsg:4326"))
> proj4string(NY_ll)

[1] " +init=epsg:4326 +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"

> writeOGR(NY_ll, dsn=paste(td, "NY.kml", sep="/"), layer="NY", driver="KML")
```

To write the tract boundaries as a KML file, we need to transform it to geographical coordinates using the appropriate `spTransform` method for `SpatialPolygons` objects, and employing lookup in the EPSG table to define the target coordinate reference system. We use `writeOGR` to write to a file, specifying the required driver. Naturally, without the linked open source GDAL, PROJ.4 and Expat libraries, the programming involved would be much more demanding, probably prohibitively so, should one wish to access many different data formats. For example, GDAL includes an OGR WFS driver:

```
> ogrInfo("WFS:http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap", "popplace")

Source: "WFS:http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap", layer: "popplace"
Driver: WFS number of rows 497
Feature type: wkbPoint with 2 dimensions
+proj=lcc +lat_1=49 +lat_2=77 +lat_0=49 +lon_0=-95 +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs
Number of fields: 1
    name type length typeName
1 gml_id    4      0   String
```

The **raster** package (Hijmans and van Etten, 2011) uses **rgdal** extensively to manage access to tiles of raster data. It is also used by the new **landsat** package documented by Goslee (2011), which is intended to support research into atmospheric and topographic correction methods for multispectral satellite data. Another interesting package using **rgdal** is **aqp** providing algorithms related to modelling of soil resources, soil classification, soil profile aggregation, and visualization (Beaudette and Roudier, 2011).

4.3 Geometry Engine, Open Source — rgeos

Development of the GEOS library interface to R began in late 2009, and made much progress in the 2010 Google Summer of Code, with Colin Rundel making a large contribution. The **rgeos** package was released on CRAN in March 2011 (Bivand and Rundel, 2011), and is beginning to be used in other packages. A Windows binary package for both architectures is available on CRAN, and for OSX on CRAN Extras, thanks to Brian Ripley and Uwe Ligges. The interface is programmed using the GEOS C API, and uses the thread-safe handle offered by GEOS. One issue uncovered by Colin Rundel in his work on the interface was the importance of the coordinate precision model, which can now be manipulated from R using `setScale`.

So far, many of the predicates and operators are applied to all member geometries, but work is progressing, spurred by clear needs demonstrated by Altman and McDonald (2011) in the **BARD** — Better Automated ReDistricting — package (Altman, 2011) for finding reduced sets of candidate pairs of contiguous geometries. Using the GEOS Sort-Tile-Recursive (STR) tree, we build a tree of geometry envelopes (bounding boxes), and then query with the same envelopes with `gUnarySTRtreeQuery`, passing the output candidate neighbours to the `poly2nb` function in the **spdep** package:

```
> la_blks <- readOGR(".", "tgr06037blk00")
> library(spdep)
> library(rgeos)
> system.time(nb <- poly2nb(la_blks, foundInBox=gUnarySTRtreeQuery(la_blks)))
```

```
   user  system elapsed 
 14.723   0.111  15.167 
```

```
> nb
```

```
Neighbour list object:
Number of regions: 89614 
Number of nonzero links: 623984 
Percentage nonzero weights: 0.007770013 
Average number of links: 6.963019 
```

Finding the neighbours is very much faster than using the internal brute-force approach for finding overlapping bounding boxes in `poly2nb`, and working with all the census blocks in Los Angeles becomes feasible. The new **pycno** package by Brunsdon (2011) also uses **rgeos** internally for pycnophylactic interpolation; here we smooth the leukemia rate from the 8 county NY data set. In addition, for display, we use `gBuffer` from **rgeos** to add a 10km buffer around supposed point source pollution sites shown in Figure 4:

```
> library(pycno)
```

```
> NY$eZ <- (1000*(NY$TRACTCAS+1))/NY$POP8
```

```
> NYp <- pycno(NY, pops=NY$eZ, celldim=500)
```

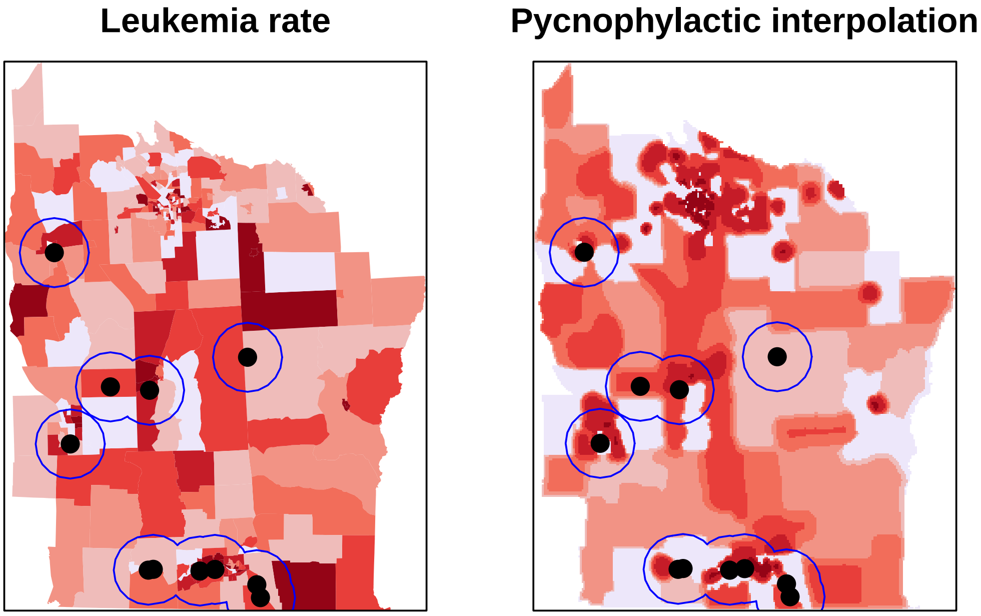

Figure 4: Choropleth and pycnophylactic interpolation maps of the leukemia rate 1978–1982, NY state 8 county data set (note that figure class intervals are not aligned with each other)

```
> TCE <- readOGR(dsn=td, layer="TCE")
> TCE10k <- gBuffer(TCE, width=10000)
```

Work on **rgeos** is continuing actively, and improvements in stability and speed can be expected as more users report their experiences. 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 assignation of interior rings to exterior rings. Had the `SpatialPolygons` class in **sp** been designed a little later, it might well have followed the OGC SFS geometry specification, but this in turn would have led to additional difficulties for users without conformant data.

4.4 Geographic Resources Analysis Support System — spgrass6

The original interface package between GRASS 5 and R, **GRASS**, written and released on CRAN in 2000, was tight-coupled, including a local copy of the core GRASS library, so that GRASS database files could be read into and written from R (Bivand, 2000; Bivand and Neteler, 2000). Figure 5 shows how the R session was started from the command prompt in a running GRASS session and location, giving ready access to GRASS and other commands through `system`, and to GRASS itself through calls to compiled C functions. This had the advantage of speed, but the weakness of containing a modified fork of the core GRASS library, modified to use the R error handler and to remove all calls to `exit` in GRASS C code. Merging in revisions from the GRASS trunk was demanding, but the interface served for five years, supporting among others Grohmann (2004) in work on morphometric analysis, and Garzón et al. (2006) on predicting habitat suitability.

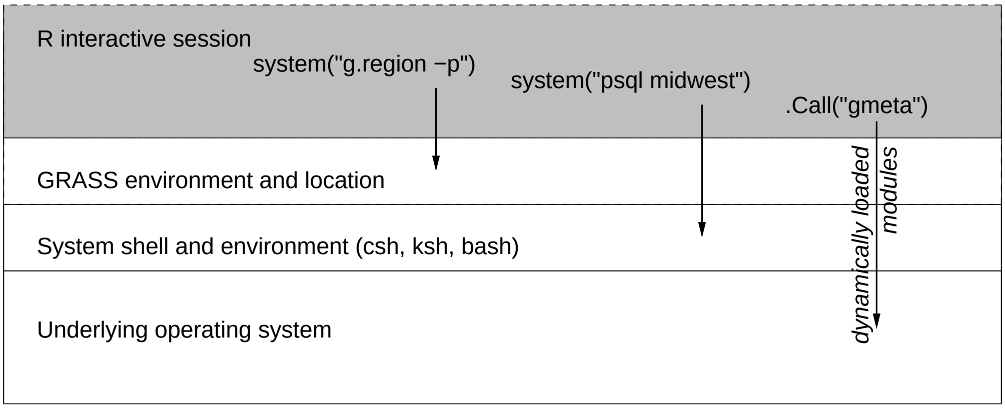
R interactive session
system("g.region −p")
system("psql midwest")
.Call("gmeta")
GRASS environment and location
System shell and environment (csh, ksh, bash)
Underlying operating system
dynamically loaded modules

Figure 5: Design of software layering in the GRASS 5 interface to R, using compiled C code.

Figure 6, showing the relative positions of software components, has two readings: the continuous line bounded box represents a setting similar to that of the **GRASS** package, with R facing the users. The second reading is the larger dashed box, where R is a computational software component in a larger system, with a GIS or other (web) application facing the users, probably through a GUI. The application then uses R, thought of as running within a GIS session and location, to handle data from storage for visualization and/or storage.

GRASS 6 was released in March 2005, and has now reached 6.4.1, with 6.4.2 imminent (GRASS Development Team, 2011). Bivand et al. (2008, pp. 99–106) and Neteler and Mitasova (2008, pp. 353–364) describe the use of the re-implemented interface package **spgrass6**, which works with GRASS 6 and the development version GRASS 7. The interface was released from a Sourceforge project at the same time as GRASS 6, and was accepted on CRAN in August 2006, to be ready for a workshop at the first FOSS4G OSGeo conference in Lausanne, Switzerland. **spgrass6** is loose-coupled, using GDAL on both sides of the interface to exchange

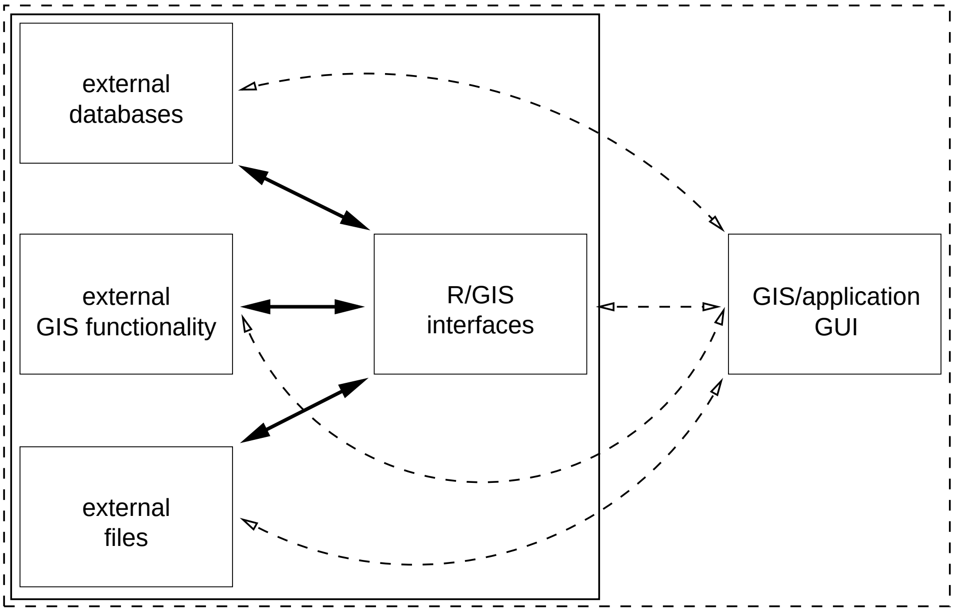

Figure 6: Positioning of software components involving R and GIS.

vector and raster data by writing to and reading from a temporary directory. If a GRASS-GDAL plugin is present, data can be read directly from GRASS into R using GDAL, but using temporary files is robust and not very wasteful of time or disk space.

From April 2009, **spgrass6** was revised to support a second mode of operation (Bivand, 2011). 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, and if necessary creating a throw-away location for use until the termination of the R session. 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, and other attributes in an XML file. Using the **XML** package in R to parse the interface decsriptions made it possible to write the functions:`parseGRASS` to parse the interface once for each GRASS command used, caching the results; `doGRASS` to collate a string comprising the GRASS command and user-supplied flags and parameters, all checked against the interface description; and `execGRASS` to run the command through `system` in a portable way. The arguments to the two latter functions have recently been simplified thanks to suggestions from Rainer Krug.

In the first example, we initiate a GRASS session from R, using the `SpatialGridDataFrame` object from `pycno` as a location template. Next we export a `SpatialPolygonsDataFrame` object to GRASS with `writeVECT6` and apply a helper function `vect2neigh` written by Markus Neteler, exploiting the topological vector representation in GRASS to return rook neighbours (with non-zero length shared boundaries) with shared boundary length in metres and total boundary length per feature, with features identified by their GRASS category numbers in this case:

```
> library(spgrass6)
> set.ignore.stderrOption(TRUE)
> initGRASS("/home/rsb/topics/grass/g642/grass-6.4.2svn", home=tempdir(), SG=NYp)
> writeVECT6(NY, vname="NY", v.in.ogr_flags="o")
> bl <- vect2neigh("NY", ID="cat", units="me")
```

```
> str(bl)
```

```
Classes ‘GRASSneigh’, ‘spatial.neighbour’ and 'data.frame':       1536 obs. of  3 variables:
 $ left  : int  1 1 1 1 1 1 1 2 2 2 ...
 $ right : int  2 13 14 15 48 49 50 1 3 13 ...
 $ length: num  732 902 458 1804 145 ...
 - attr(*, "external")= num  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "total")= Named num  1329 5178 5620 13156 5139 ...
  ..- attr(*, "names")= chr  "-1" "1" "2" "3" ...
 - attr(*, "region.id")= chr  "1" "2" "3" "4" ...
 - attr(*, "n")= int 281
```

The second example replicates the **rgeos** `gBuffer` above, by exporting a `SpatialPointsDataFra` object to GRASS with `writeVECT6`, and using `execGRASS` to run the GRASS command `v.buffer` on the input vector object, returning the results to R with `readVECT6`. The use of the “6” tag in **spgrass6** function names is now misleading, as the functions work for GRASS versions 6 and 7, but was originally introduced to signal the difference from GRASS version 5. Generic wrappers will be put in place before GRASS 7 is released, and the package name will be modified to suit.

```
> writeVECT6(TCE, vname="TCE", v.in.ogr_flags="o")
> execGRASS("v.buffer", input="TCE", output="TCE10k", distance=10000)
> TCE10kG <- readVECT6("TCE10k", with_c=TRUE)
```

The output buffers from GRASS `v.buffer` and **rgeos** `gBuffer` are not exactly identical, because they do not use the same numbers of boundary coordinates in the same positions to represent the buffers. Overplotting does, however, show that the buffers are the same given those representational differences.

The interface between GRASS 6 and R has been used in research in a number of fields, for example by Carrera-Hernández and Gaskin (2008) in implementing the Basin of Mexico hydrogeological database, and by Grohmann and Steiner (2008) in SRTM resampling using short distance kriging. The work by Haywood and Stone (2011) is interesting in that it uses the interface to apply the Weka machine learning software suite, itself interfaced to R through the **RWeka** package, to GIS data in GRASS; R then becomes a convenient bridge between applications, with the GRASS–R interface opening up other possibilities beyond R. Finally, Jasiewicz (2011) reports the transfer of fuzzy inference system technology to a GRASS add-on after prototyping using an R implementation in the **sets** package (Meyer and Hornik, 2009), which was not intended for large data sets.

4.5 SAGA — RSAGA, Geoprocessing — RPyGeo, MGET — Marine Geospatial Ecology Tools and others

The open source SAGA GIS has been interfaced with R using the SAGA command line interface in the **RSAGA** package first released to CRAN in early 2008 (Brenning, 2011). The package provides extensive facilities for scripting SAGA through R, as **spgrass6** also now does, since using R to script repetitive tasks in a GIS turned out to be of value to researchers. The author of **RSAGA**, Alexander Brenning, has also contributed the **RPyGeo** package, based on **RSAGA**, to autogenerate and run ArcGIS Python Geoprocessor commands from R. Further examples of the use of R with ArcGIS are given by Krivoruchko (2011) to supplement methods implemented in ArcGIS, and by Roberts et al. (2010), who have developed Marine Geospatial Ecology Tools (MGET, also known as the GeoEco Python package).[39]

The R interface with SAGA has been used by Brenning (2009) for integrating terrain analysis and multispectral remote sensing in automatic rock glacier detection, using modern regression techniques — the availability of many varied techniques in R permitted them to be evaluated rapidly. Goetz et al. (2011) follow this up in integrating physical and empirical landslide models. In a paper on geostatistical modelling of topography, Hengl et al. (2008) uses the interface between R and SAGA to benefit from the strengths of both software components. Hengl et al. (2010) addresses the associated problem of stream network uncertainty, when the stream networks are derived from interpolated elevation data, again using the interface between SAGA and R; R is also used extensively for scripting SAGA. Tomislav Hengl is also very active in organising courses for field scientists using open source geospatial software, especially the GEOSTAT series,[40] run in the open source spirit, and now with a useful collection of video recordings.

Finally, it is worth noting that TerraLib is linked to R using **sp** classes in the **aRT** package, which uses R as the computational front-end and TerraLib as a data store and middleware component.[41]

5 Future prospects

We have seen above that TerraLib, presented by Câmara et al. (2008) and Câmara et al. (2010), has offered broad functionality, and excellent support for research, exemplified by de Espindola et al. (2011). The current version, 4.1.0, is well supported,

[39] `http://code.env.duke.edu/projects/mget`.

[40] `http://geostat-course.org/`.

[41] `http://www.leg.ufpr.br/aRT`.

but the TerraLib developers are moving to embrace much more of the OSGeo community than in the past.

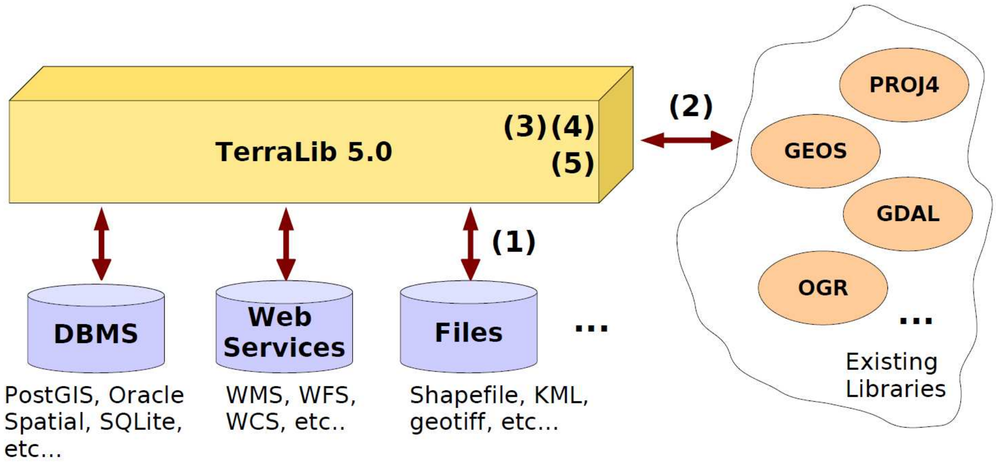

Figure 7: Envisioned developments in TerraLib 5: (1) Support to different kinds of data sources; (2) Extensive use of existing libraries; (3) More modular, simpler and more easily extensible architecture; (4) OGC compliant (SFS-SQL, OGC Web Service, ...); (5) Represent and query spatio-temporal data

Figure 7 is taken from a presentation[42] by Karine Reis Ferreira and Pedro Ribeiro Andrade, and reflects some of the prospects being considered by the development team for release by 2013. These design choices engage many more existing projects, leveraging the user and developer communities of these projects, and increasing opportunities for shared technology exploration and development. 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, and how the **aRT** interface with R progresses.

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. I believe that mailing lists have a hidden bonus, that is that readers, if they are willing to do so, can read threads that are not relevant to their current concerns, but which may offer insight that will increase

[42] `http://giv-wikis.uni-muenster.de/agp/pub/Main/SpatioTemporalDataInRWorkshop2011/`

productivity later on.[43]

Naturally, other forms of communication, such as IRC chats in OSGeo, or blogs by Barry Rowlingson[44] and Daniel Nüst[45] are interesting, and less serious — more encouraging — in tone; being able to include graphics in wiki pages, web fora and blogs is often very helpful. Being able to search these resources and mailing list archives remains important, and may not be easy to achieve. So 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.

Whether memory of past achievements and insights is a future prospect or not is, of course, a rather paradoxical issue. I would argue that very much of what geocomputation has been doing over the past decades has been to compute on, and develop computational techniques for handling, problems that were proposed in earlier years, when analysis of even small data sets was exhausting (Bivand, 2009). In that sense, memory is not unimportant, because we may make progress when we confront the insights and propositions of the past with regard to research problems, algorithms, or tools with fresh opportunities offered by advances in data gathering and computational technologies, both in hardware and in software. In this context, 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

Altman, M. (2011). Bard: Better automated redistricting. R package version 1.24, `http://CRAN.R-project.org/package=BARD`.

Altman, M. and McDonald, M. P. (2011). BARD: Better automated redistricting. *Journal of Statistical Software*, 42(4):1–28.

Batcheller, J. and Reitsma, F. (2010). Implementing feature level semantics for spatial data discovery: Supporting the reuse of legacy data using open source components. *Computers, Environment and Urban Systems*, 34(4):333–344.

Beaudette, D. and Roudier, P. (2011). aqp: Algorithms for quantitative pedology. R package version 0.99-5, `http://CRAN.R-project.org/package=aqp`.

Bentley, J., Knuth, D. E., and McIlroy, M. D. (1986). Programming pearls: a literate program. *Communications of the ACM*, 29(6):471–483.

[43] I read the development lists of R, GRASS, GDAL, GEOS and PROJ.4, among others, and many of the insights presented in this chapter have matured from exchanges on these lists; administering the R-sig-geo list has been a most valuable source of information.

[44] `http://geospaced.blogspot.com/`.

[45] `http://www.nordholmen.net/sos4r/`.

Bivand, R. (2000). Using the r statistical data analysis language on grass 5.0 gis database files. *Computers & Geosciences*, 26(9–10):1043–1052.

Bivand, R. (2009). Applying Measures of Spatial Autocorrelation: Computation and Simulation. *Geographical Analysis*, 41(4):375–384.

Bivand, R. (2011). spgrass6: Interface between grass 6 and r. R package version 0.7-4, `http://CRAN.R-project.org/package=spgrass6`.

Bivand, R. and Lucas, A. E. (2000). Integrating models and geographical information systems. In Openshaw, S. and Abrahart, R., editors, *GeoComputation*, pages 331–363. Taylor & Francis, London.

Bivand, R. and Neteler, M. (2000). Open source geocomputation: Using the R data analysis language integrated with GRASS GIS and PostgreSQL data base systems. In *GeoComputation 2000*. Proceedings of the 5th International Conference on GeoComputation. `http://spatial.nhh.no/gc00/geocomp2000.pdf`.

Bivand, R., Pebesma, E. J., and Gómez-Rubio, V. (2008). *Applied Spatial Data Analysis with R*. Springer, New York.

Bivand, R. and Rundel, C. (2011). rgeos: Interface to geometry engine - open source (geos). R package version 0.1-9, `http://CRAN.R-project.org/package=rgeos`.

Bray, R. (2008). MapGuide open source. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 131–152. Springer-Verlag, Berlin.

Brenning, A. (2009). Benchmarking classifiers to optimally integrate terrain analysis and multispectral remote sensing in automatic rock glacier detection. *Remote Sensing of Environment*, 113(1):239–247.

Brenning, A. (2011). Rsaga: Saga geoprocessing and terrain analysis in r. R package version 0.92-1, `http://CRAN.R-project.org/package=RSAGA`.

Brunsdon, C. (2011). pycno: Pycnophylactic interpolation. R package version 1.1, `http://CRAN.R-project.org/package=pycno`.

Cagnacci, F. and Urbano, F. (2008). Managing wildlife: A spatial information system for gps collars data. *Environmental Modelling & Software*, 23(7):957–959.

Carrera-Hernández, J. and Gaskin, S. (2008). The basin of mexico hydrogeological database (BMHDB): Implementation, queries and interaction with open source software. *Environmental Modelling & Software*, 23(10–11):1271–1279.

Chambers, J. M. (2008). *Software for Data Analysis: Programming with R*. Springer, New York.

Chen, D., Shams, S., Carmona-Moreno, C., and Leone, A. (2010). Assessment of open source gis software for water resources management in developing countries. *Journal of Hydro-environment Research*, 4(3):253–264.

Chen, R. and Xie, J. (2008). Open source databases and their spatial extensions. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 105–129. Springer-Verlag, Berlin.

Cressie, N. and Wikle, C. K. (2011). *Statistics for spatio-temporal data*. John Wiley, Hoboken, NJ.

Câmara, G., Vinhas, L., and de Souza, R. C. M. (2010). Free and open source GIS: Will there ever be a geo-linux? In Bocher, E., editor, *Proceedings of Open Source Geospatial Research Conference (OGRS 2009)*, Lecture Notes in Geoinformation and Cartography. Springer, Berlin Heidelberg.

Câmara, G., Vinhas, L., Ferreira, K. R., de Queiroz, G. R., de Souza, R. C. M., Monteiro, A. M. V., de Carvalho, M. T., Casanova, M. A., and de Freitas, U. M. (2008). Terralib: An open source GIS library for large-scale environmental and socio-economic applications. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 247–270. Springer-Verlag, Berlin.

de Espindola, G. M., de Aguiar, A. P. D., Pebesma, E., Câmara, G., and Fonseca, L. (2011). Agricultural land use dynamics in the brazilian amazon based on remote sensing and census data. *Applied Geography*, 32(2):240–252.

Dunfey, R., Gittings, B., and Batcheller, J. (2006). Towards an open architecture for vector gis. *Computers & Geosciences*, 32(10):1720–1732.

Eddelbuettel, D. and Francois, R. (2011). Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18.

Erle, S., Gibson, R., and Walsh, J. (2005). *Mapping Hacks: Tips & Tools for Electronic Cartography*. O'Reilly, Sebastopol, CA.

Fisher, R. and Myers, B. (2011). Free and simple gis as appropriate for health mapping in a low resource setting: a case study in eastern indonesia. *International Journal of Health Geographics*, 10(15):1–11.

Fox, J. (2009). Aspects of the Social Organization and Trajectory of the R Project. *The R Journal*, 1(2):5–13.

Gahegan, M., Hardisty, F., Demšar, U., and Takatsuka, M. (2008). GeoVISTA Studio: Reusability by design. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 201–220. Springer-Verlag, Berlin.

Garzón, M. B., Blazek, R., Neteler, M., de Diosa, R. S., Olleroa, H. S., and Furlanello, C. (2006). Predicting habitat suitability with machine learning models: The potential area of *Pinus sylvestris* l. in the iberian peninsula. *Ecological Modelling*, 197(3–4):383–393.

Gibson, R. and Erle, S. (2006). *Google Maps Hacks*. O'Reilly, Sebastopol, CA.

Goetz, J., Guthrie, R., and Brenning, A. (2011). Integrating physical and empirical landslide susceptibility models using generalized additive models. *Geomorphology*, 129(3–4):376–386.

Goslee, S. (2011). Analyzing remote sensing data in r: The landsat package. *Journal of Statistical Software*, 43(4):1–25.

GRASS Development Team (2011). GRASS 6.4 Users Manual. `http://grass.osgeo.org/grass64/manuals/html64_user/`.

Grohmann, C. (2004). Morphometric analysis in geographic information systems: applications of free software GRASS and R. *Computers & Geosciences*, 30(9–10):1055–1067.

Grohmann, C. and Steiner, S. (2008). Srtm resample with short distance-low nugget kriging. *International Journal of Geographical Information Science*, 22(8):895–906.

Hall, G. B., Chipeniuk, R., Feick, R. D., Leahy, M. G., and Deparday, V. (2010). Community-based production of geographic information using open source software and web 2.0. *International Journal of Geographical Information Science*, 24(5):761–781.

Hall, G. B. and Leahy, M. (2008a). Design and implementation of a map-centred synchronous collaboration tool using open source components: The MapChat project. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 221–245. Springer-Verlag, Berlin.

Hall, G. B. and Leahy, M., editors (2008b). *Open Source Approaches in Spatial Data Handling*. Springer-Verlag, Berlin.

Haywood, A. and Stone, C. (2011). Mapping eucalypt forest susceptible to dieback associated with bell miners *(Manorina melanophys)* using laser scanning, spot 5 and ancillary topographical data. *Ecological Modelling*, 222(5):1174–1184.

Hengl, T., Bajat, B., Blagojević, D., and Reuter, H. (2008). Geostatistical modeling of topography using auxiliary maps. *Computers & Geosciences*, 34(12):1886–1899.

Hengl, T., Heuvelink, G., and van Loon, E. (2010). On the uncertainty of stream networks derived from elevation data: the error propagation approach. *Hydrology and Earth System Sciences*, 14(7):1153–1165.

Hijmans, R. J. and van Etten, J. (2011). raster: Geographic analysis and modeling with raster data. R package version 1.9-11, `http://CRAN.R-project.org/package=raster`.

Jasiewicz, J. (2011). A new GRASS GIS fuzzy inference system for massive data analysis. *Computers & Geosciences*, 37(9):1525–1531.

Jasiewicz, J. and Metz, M. (2011). A new GRASS GIS toolkit for Hortonian analysis of drainage networks. *Computers & Geosciences*, 37(8):1162–1173.

Keitt, T. H., Bivand, R., Pebesma, E., and Rowlingson, B. (2011). rgdal: Bindings for the geospatial data abstraction library. R package version 0.7-2, `http://CRAN.R-project.org/package=rgdal`.

Kernighan, B. W. and Pike, R. (1984). *The UNIX programming environment*. Prentice-Hall, Englewood Cliffs, N. J.

Kernighan, B. W. and Pike, R. (1999). *The practice of programming*. Addison-Wesley, Reading, Mass.

Kernighan, B. W. and Plauger, P. J. (1976). *Software tools*. Addison-Wesley, Reading, Mass.

Kralidis, A. T. (2008). Geospatial open source and open standards convergences. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 1–20. Springer-Verlag, Berlin.

Krivoruchko, K. (2011). *Spatial Statistical Data Analysis for GIS Users*. ESRI Press, Redlands, CA. DVD.

Krug, R. M., Roura-Pascual, N., and Richardson, D. M. (2010). Clearing of invasive alien plants under different budget scenarios: using a simulation model to test efficiency. *Biological Invasions*, 12(12):4099–4112.

Leisch, F. and Rossini, A. J. (2003). Reproducible statistical research. *Chance*, 16(2):46–50.

Lewin-Koh, N. J., Bivand, R., and others (2011). maptools: Tools for reading and handling spatial objects. R package version 0.8-10, `http://CRAN.R-project.org/package=maptools`.

Lime, S. (2008). MapServer. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 65–85. Springer-Verlag, Berlin.

Luis, J. F. (2007). Mirone: A multi-purpose tool for exploring grid data. *Computers & Geosciences*, 33(1):31–41.

McIhagga, D. (2008). Communities of practice and the business of open sourceweb mapping. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 49–64. Springer-Verlag, Berlin.

Meyer, D. and Hornik, K. (2009). Generalized and customizable sets in r. *Journal of Statistical Software*, 31(2):1–27.

Mitchell, T. (2005). *Web Mapping Illustrated*. O'Reilly, Sebastopol, CA.

Neteler, M., Beaudette, D., Cavallini, P., Lami, L., and Cepicky, J. (2008). GRASS GIS. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 171–199. Springer-Verlag, Berlin.

Neteler, M. and Mitasova, H. (2008). *Open Source GIS: A GRASS GIS Approach, Third Edition*. Springer, New York.

Nüst, D., Stasch, C., and Pebesma, E. (2011). Connecting r to the sensor web. In Geertman, S., Reinhardt, W., and Toppen, F., editors, *Advancing Geoinformation Science for a Changing World*, volume 1 of *Lecture Notes in Geoinformation and Cartography*, pages 227–246. Springer, Berlin Heidelberg.

Pebesma, E., Bivand, R., and others (2011a). sp: classes and methods for spatial data. R package version 0.9-88, `http://CRAN.R-project.org/package=sp`.

Pebesma, E., Cornford, D., Dubois, G., Heuvelink, G. B. M., Hristopulos, D., Pilz, J., Stoehlker, U., Morin, G., and Skoien, J. O. (2011b). INTAMAP: The design and implementation of an interoperable automated interpolation web service. *Computers & Geosciences*, 37(3):343–352.

R Development Core Team (2011). R: A language and environment for statistical computing. ISBN 3-900051-07-0, `http://www.R-project.org/`.

Ramsey, P. (2007). The state of open source GIS. Technical report, Refractions Research Inc.

Rey, S. J. (2009). Show me the code: spatial analysis and open source. *Journal of Geographical Systems*, 11(2):191–207.

Roberts, J. J., Best, B. D., Dunn, D. C., Treml, E. A., and Halpin, P. N. (2010). Marine geospatial ecology tools: An integrated framework for ecological geoprocessing with arcgis, python, r, matlab, and c++. *Environmental Modelling & Software*, 25(10):1197–1207.

Robertson, C. and Farmer, C. J. Q. (2008). Developing an open-source framework for surveillance and analysis of emerging zoonotic diseases. In *Proceedings of the Fifth National Symposium on Geo-Informatics*, pages 123–134, Colombo, Sri Lanka. Geo-Informatics Society of Sri Lanka.

Robertson, C., Farmer, C. J. Q., Nelson, T. A., Mackenzie, I. K., Wulder, M. A., and White, J. C. (2009). Determination of the compositional change (1999–2006) in the pine forests of british columbia due to mountain pine beetle infestation. *Environmental Monitoring and Assessment*, 158(1–4):593–608.

Rocchini, D., Metz, M., Frigeri, A., Delucchi, L., Marcantonio, M., and Neteler, M. (2011). Robust rectification of aerial photographs in an open source environment. *Computers & Geosciences*.

Roiz, D., Neteler, M., Castellani, C., Arnoldi, D., and Rizzoli, A. (6). Climatic factors driving invasion of the tiger mosquito *(Aedes albopictus)* into new areas of Trentino, northern Italy. *PLoS ONE*, 4:e14800.

Roura-Pascual, N., Richardson, D. M., Krug, R. M., Brown, A., Chapman, R. A., Forsyth, G. G., Maitre, D. C. L., Robertson, M. P., Stafford, L., Wilgen, B. W. V., Wannenburgh, A., and Wessels, N. (2009). Ecology and management of alien plant invasions in South African fynbos: Accommodating key complexities in objective decision making. *Biological Conservation*, 142(8):1595–1604.

Schweik, C. M., Fernandez, M. T., Hamel, M. P., Kashwan, P., Lewis, Q., and Stepanov, A. (2009). Reflections of an online geographic information systems course based on open source software. *Social Science Computer Review*, 27(1):118–129.

Sherman, G. E. (2008). *Desktop GIS: Mapping the Planet with Open Source Tools*. Pragmatic Bookshelf, http://pragprog.com/.

Sorokine, A. (2007). Implementation of a parallel high-performance visualization technique in GRASS GIS. *Computers & Geosciences*, 33(5):685–695.

Steiniger, S. and Bocher, E. (2009). An overview on current free and open source desktop gis developments. *International Journal of Geographical Information Science*, 23(10):1345–1370.

Steiniger, S. and Hay, G. J. (2009). Free and open source geographic information tools for landscape ecology. *Ecological Informatics*, 4(4):183–195.

Sui, D. and DeLyser, D. (2011). Crossing the qualitative-quantitative chasm i: Hybrid geographies, the spatial turn, and volunteered geographic information (vgi). *Progress in Human Geography*.

Theußl, S., Ligges, U., and Hornik, K. (2011). Prospects and challenges in r package development. *Computational Statistics*, 26(3):395–404.

Turton, I. (2008). GeoTools. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 153–169. Springer-Verlag, Berlin.

van Etten, J. and Hijmans, R. J. (2010). A geospatial modelling approach integrating archaeobotany and genetics to trace the origin and dispersal of domesticated plants. *PLoS ONE*, 5(8):e12060.

Vanmeulebrouk, B., Rivett, U., Ricketts, A., and Loudon, M. (2008). Open source gis for hiv/aids management. *International Journal of Health Geographics*, 7(53):1–16.

Waller, L. A. and Gotway, C. A. (2004). *Applied Spatial Statistics for Public Health Data*. John Wiley & Sons, Hoboken, NJ.

Walter, G., Warmerdam, F., and Farris-Manning, P. (2002). An open source tool for geospatial image exploitation. In *IGARSS 2002: Integrating our view of the planet*, IEEE International Symposium on Geoscience and Remote Sensing (IGARSS), pages 3522–3524. IEEE.

Warmerdam, F. (2008). The geospatial data abstraction library. In Hall, G. B. and Leahy, M., editors, *Open Source Approaches in Spatial Data Handling*, pages 87–104. Springer-Verlag, Berlin.

Weidmann, N. B. and Gleditsch, K. S. (2010). Mapping and Measuring Country Shapes. *The R Journal*, 2(1):18–24.

Weidmann, N. B., Kuse, D., and Gleditsch, K. S. (2011). cshapes: Cshapes dataset and utilities. R package version 0.3-1, `http://CRAN.R-project.org/package=cshapes`.

Yi, Q., Hoskins, R. E., Hillringhouse, E. A., Sorensen, S. S., Oberle, M. W., Fuller, S. S., and Wallace, J. C. (2008). Integrating open-source technologies to build low-cost information systems for improved access to public health data. *International Journal of Health Geographics*, 7(29):1–13.

Issued in the series Discussion Papers 2010

2010

01/10 January, Øystein Foros, **Hans Jarle Kind**, and **Greg Shaffer**, "Mergers and Partial Ownership"

02/10 January, **Astrid Kunze** and Kenneth R. Troske, "Life-cycle patterns in male/female differences in job search".

03/10 January, **Øystein Daljord** and **Lars Sørgard**, "Single-Product versus Uniform SSNIPs".

04/10 January, **Alexander W. Cappelen**, James Konow, **Erik Ø. Sørensen**, and **Bertil Tungodden**, "Just luck: an experimental study of risk taking and fairness".

05/10 February, **Laurence Jacquet**, "Optimal labor income taxation under maximin: an upper bound".

06/10 February, **Ingvild Almås**, Tarjei Havnes, and Magne Mogstad, "Baby booming inequality? Demographic change and inequality in Norway, 1967-2004".

07/10 February, **Laurence Jacquet**, Etienne Lehmann, and Bruno van der Linden, "Optimal redistributive taxation with both extensive and intensive responses".

08/10 February, **Fred Schroyen**, "Income risk aversion with quantity constraints".

09/10 March, **Ingvild Almås** and Magne Mogstad, "Older or Wealthier? The impact of age adjustment on cross-sectional inequality measures".

10/10 March, Ari Hyytinen, **Frode Steen**, and Otto Toivanen, "Cartels Uncovered".

11/10 April, **Karl Ove Aarbu**, "Demand patterns for treatment insurance in Norway".

12/10 May, **Sandra E. Black**, Paul J. Devereux, and **Kjell G. Salvanes**, "Under pressure? The effect of peers on outcomes of young adults".

13/10 May, **Ola Honningdal Grytten** and Arngrim Hunnes, "A chronology of financial crises for Norway".

14/10 May, Anders Bjørklund and **Kjell G. Salvanes**, "Education and family background: Mechanisms and policies".

15/10 July, **Eva Benedicte D. Norman** and **Victor D. Norman**, "Agglomeration, tax competition and local public goods supply".

16/10 July, **Eva Benedicte D. Norman**, "The price of decentralization".

17/10 July, **Eva Benedicte D. Norman**, "Public goods production and private sector productivity".

18/10 July, **Kurt Richard Brekke**, Tor Helge Holmås, and Odd Rune Straume, "Margins and Market Shares: Pharmacy Incentives for Generic Substitution".

19/10 August, **Karl Ove Aarbu**, "Asymmetric information – evidence from the home insurance market".

20/10 August. **Roger Bivand**, "Computing the Jacobian in spatial models: an applied survey".

21/10 August, **Sturla Furunes Kvamsdal**, "An overview of Empirical Analysis of behavior of fishermen facing new regulations.

22/10 September, Torbjørn Hægeland, Lars Johannessen Kirkebøen, Odbjørn Raaum, and **Kjell G. Salvanes**, " Why children of college graduates outperform their schoolmates: A study of cousins and adoptees".

23/10 September**, Agnar Sandmo**, "Atmospheric Externalities and Environmental Taxation".

24/10 October, **Kjell G. Salvanes,** Katrine Løken, and Pedro Carneiro**, "**A flying start? Long term consequences of maternal time investments in children during their first year of life".

25/10 September, **Roger Bivand, "**Exploiting Parallelization in Spatial Statistics: an Applied Survey using R".

26/10 September, **Roger Bivand**, "Comparing estimation methods for spatial econometrics techniques using R".

27/10 October. **Lars Mathiesen**, **Øivind Anti Nilsen**, and **Lars Sørgard**, "Merger simulations with observed diversion ratios."

28/10 November, **Alexander W. Cappelen, Knut Nygaard, Erik Ø. Sørensen**, and **Bertil Tungodden**, "Efficiency, equality and reciprocity in social preferences: A comparison of students and a representative population".

29/10 December, **Magne Krogstad Asphjell**, Wilko Letterie, **Øivind A. Nilsen**, and Gerard A. Pfann, ”Sequentiality versus Simultaneity: Interrelated Factor Demand”.

2011

01/11 January, **Lars Ivar Oppedal Berge**, **Kjetil Bjorvatn**, and **Bertil Tungodden**, "Human and financial capital for microenterprise development: Evidence from a field and lab experiment."

02/11 February, **Kurt R. Brekke**, Luigi Siciliani, and Odd Rune Straume, "Quality competition with profit constraints: do non-profit firms provide higher quality than for-profit firms?"

03/11 February, **Gernot Doppelhofer** and Melvyn Weeks, "Robust Growth Determinants".

04/11 February, Manudeep Bhuller, Magne Mogstad, and **Kjell G. Salvanes**, "Life-Cycle Bias and the Returns to Schooling in Current and Lifetime Earnings".

05/11 March, **Knut Nygaard**, "Forced board changes: Evidence from Norway".

06/11 March, **Sigbjørn Birkeland d.y**., "Negotiation under possible third party settlement".

07/11 April, **Fred Schroyen**, "Attitudes towards income risk in the presence of quantity constraints".

08/11 April, Craig Brett and **Laurence Jacquet**, "Workforce or Workfare?"

09/11 May, **Bjørn Basberg**, "A Crisis that Never Came. The Decline of the European Antarctic Whaling Industry in the 1950s and -60s".

10/11 June, Joseph A. Clougherty, Klaus Gugler, and **Lars Sørgard**, "Cross-Border Mergers and Domestic Wages: Integrating Positive 'Spillover' Effects and Negative 'Bargaining' Effects".

11/11 July, **Øivind A. Nilsen**, Arvid Raknerud, and Terje Skjerpen, "Using the Helmert-transformation to reduce dimensionality in a mixed model: Application to a wage equation with worker and …rm heterogeneity".

12/11 July, Karin Monstad, Carol Propper, and **Kjell G. Salvanes**, "Is teenage motherhood contagious? Evidence from a Natural Experiment".

13/11 August, **Kurt R. Brekke**, Rosella Levaggi, Luigi Siciliani, and Odd Rune Straume, "Patient Mobility, Health Care Quality and Welfare".

14/11 July, **Sigbjørn Birkeland d.y**., "Fairness motivation in bargaining".

15/11 September, **Sigbjørn Birkeland d.y**, **Alexander Cappelen**, **Erik Ø. Sørensen**, and **Bertil Tungodden**, "Immoral criminals? An experimental study of social preferences among prisoners".

16/11 September, **Hans Jarle Kind**, Guttorm Schjelderup, and Frank Stähler, "Newspaper Differentiation and Investments in Journalism: The Role of Tax Policy".

17/11 **Gregory Corcos**, Massimo Del Gatto, Giordano Mion, and Gianmarco I.P. Ottaviano, "Productivity and Firm Selection: Quantifying the "New" Gains from Trade".

18/11 **Grant R. McDermott** and **Øivind Anti Nilsen**, "Electricity Prices, River Temperatures and Cooling Water Scarcity".

19/11 Pau Olivella and **Fred Schroyen**, "Multidimensional screening in a monopolistic insurance market".

20/11 **Liam Brunt**, "Property rights and economic growth: evidence from a natural experiment".

21/11 Pau Olivella and **Fred Schroyen**, "Multidimensional screening in a monopolistic insurance market: proofs".

22/11 **Roger Bivand**, "After "Raising the Bar": applied maximum likelihood estimation of families of models in spatial econometrics".

23/11 **Roger Bivand**, "Geocomputation and open source software:components and software stacks".

NHH

Norges
Handelshøyskole

Norwegian School of Economics

NHH
Helleveien 30
NO-5045 Bergen
Norway

Tlf/Tel: +47 55 95 90 00
Faks/Fax: +47 55 95 91 00
nhh.postmottak@nhh.no
www.nhh.no