

The R Development Process: status and prospects

Roger Bivand¹

26 June 2014

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

Overview

- 1 Introduction
 - Motivation: longer lives
- 2 R-spatial
 - What does use look like?
 - R-spatial — Building a community
 - R-spatial — **sp**
 - With a little help ...
- 3 What is R?
 - R is many things to many people
 - Developer communities
- 4 The R Development Process
 - R itself
 - CRAN, Bioconductor, github, others
 - Opportunities and challenges

Introduction

This talk will use observations from the development of contributed packages for spatial data analysis in R to cast light on:

- the current status of R development
- the management of contributed packages
- prospects

Particular stress will be placed on the problematic disproportions in the concerns of various user and developer communities with regard to the future capacity of our language and community of choice

Using spatial data

- Being explicit about the use of (spatial) data in R permits attention to be given to the important steps of data handling and representation — rushing to analysis often leads to problems
- Handling spatial data is never “easy”, because representational choices lead to support consequences, which qualify or undermine inferential outcomes
- Being close to the data and close to the code may give considerable insight and freedom, and permits co-working with disciplines sharing frames of understanding that are mutually comprehensible
- Just publishing is not going to be enough; we will need to be able to demonstrate how our conclusions have been reached by releasing code and data (some data is public already).

Visualizing longer lives I

A news item a year ago described work on overall premature mortality (<75) in England, linking to Public Health England (PHE) sites:

31 June 2013 Last updated at 06:29 GMT

Early deaths: Regional variations 'shocking' - Hunt

By James Gallagher
Health and welfare reporter, BBC News

The local variation in early death rates revealed in a new longer life for England in "shocking" and near-five orders to improve health, Health Secretary Jeremy Hunt has said.

Public Health England's Longer Lives website, which tracks local death rates, shows people in north-west England are at the greatest risk of dying early.

Mr Hunt said the data could be used to tackle smoking, drinking and obesity.

Labour called for a "One Nation approach" to end health inequalities.

The longer life rates are a colour system to rate areas tackling premature deaths based on the number of people under the age of 75 who died over a two year period.

Overall premature deaths



31 June 2013 Last updated at 06:29 GMT

Related Stories

Start up could let public health?

Why do the factors that longer life not?

North south health divide widening



Public Health England

Longer Lives

About the data

The data presented in Longer Lives is drawn from data published for the Public Health Outcomes Framework (PHOF).

Where the data comes from

The PHOF data is the best available data available for England. It is based on data from local authorities and is based on data from local authorities.

The data is based on the number of deaths under the age of 75 in each local authority area in England. The data is based on the number of deaths under the age of 75 in each local authority area in England. The data is based on the number of deaths under the age of 75 in each local authority area in England.

How the rates are calculated

Further details on the longer life rates are available on the website. The data is based on the number of deaths under the age of 75 in each local authority area in England. The data is based on the number of deaths under the age of 75 in each local authority area in England. The data is based on the number of deaths under the age of 75 in each local authority area in England.

What the colour key in brackets

The rates and local authority data are on a color scale from worst to best. The colors are: Worst (red), Above the average (orange), Below the average (yellow), Best (green), and Data unavailable (grey).

For any detailed data on the rates in the above table

Worst
Above the average
Below the average
Best
Data unavailable

Visualizing longer lives II

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



Roger Bivand · 3 months ago

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

^ | v Reply Share



Paul Fryers, PHE → Roger Bivand · 3 months ago

The detailed metadata for the indicators are available on the Public Health Outcomes Framework Data Tool (<http://www.phoutcomes.info>). The methodology used for the confidence intervals is described there (specifically <http://www.phoutcomes.info/pub...>), with further references for more details.

^ | v Reply Share

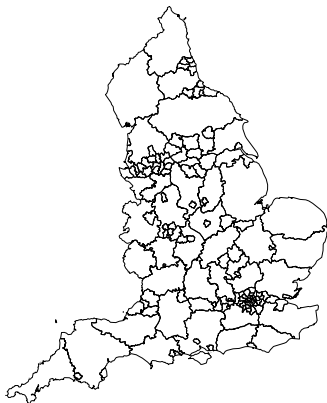
Visualizing longer lives III

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

```
> library(rgdal)
> sm <- readOGR(".", "Prem_mort_sim")

> sm1 <- sm[!is.na(sm$Value),
+          ]
> names(sm1)[16:26]
```

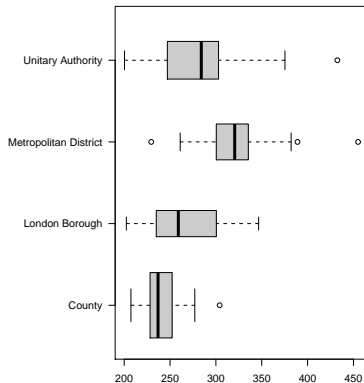
```
[1] "Indictr" "Tim_Prd" "Area_Cd"
[4] "Area_Nm" "Value"    "Lowr_CI"
[7] "Uppr_CI" "Count"   "Denmnr"
[10] "Sex"     "Age"
```



Visualizing longer lives IV

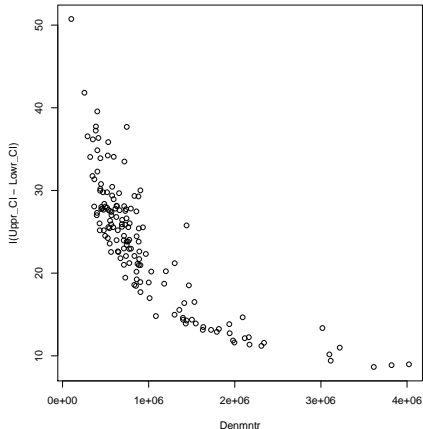
If we compare the different classes of local authorities represented in the data set for premature mortality rates per 100,000, 2009-2011, we see considerable variability between class, reflecting some of the structuring of the online maps. Covariates are also available for download, but are not used here.

Premature mortality rate per 100,000, 2009-2011, England
box width proportional to population



Visualizing longer lives V

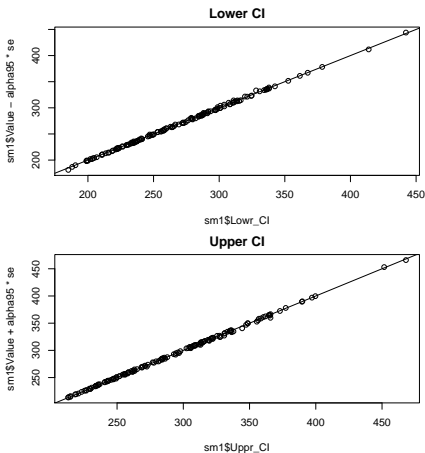
Using the data provided, and credit should be given to PHE for making it available, we can see that the gap between the upper and lower (95%) confidence intervals is large for observations with small populations and vice-versa. The Denmnr variable is three-years' worth of the age standardised European Standard Population, it turns out, and the rates (Value) are annual per 100,000:



Visualizing longer lives VI

In conversation, Paul Fryers referred to work (and spreadsheets with macros) on the Association of Public Health Observatories' website. These appear to have connections to the use of funnel plots to reflect the sample size effect on the rates from smaller aggregate areas. Using formulae from Dover and Schopflocher (2011, p. 3), we can reconstruct the confidence intervals:

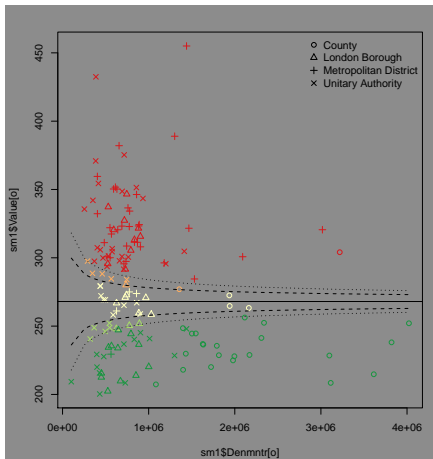
```
> alpha95 <- qnorm((1 - 0.95)/2,  
+   lower.tail = FALSE)  
> se <- sqrt((sm1$Value * (1e+05 -  
+   sm1$Value))/sm1$Denmtr)
```



Visualizing longer lives VII

Given this insight, we can construct a funnel plot and fumble towards the class intervals used for the published maps of health outcomes:

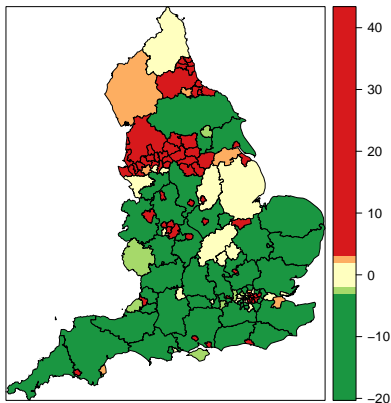
```
> alpha998 <- qnorm((1 - 0.998)/2,
+   lower.tail = FALSE)
> phat <- 268
> o <- order(sm1$Denmnr)
> se <- sqrt((phat * (1e+05 -
+   phat))/sm1$Denmnr)
> sm1$zvalue <- (sm1$Value - phat)/se
```



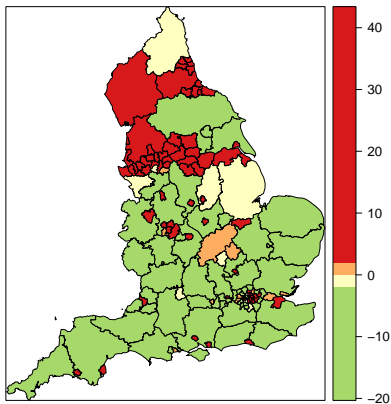
Visualizing longer lives VIII

So finally the five class 99.8% and 95% map, and the (politically chosen) four class 95% map split on zero:

Funnel plot z-scores, 5 classes



Funnel plot z-scores, 4 classes



R-spatial

- 1 Introduction
 - Motivation: longer lives
- 2 R-spatial
 - What does use look like?
 - R-spatial — Building a community
 - R-spatial — **sp**
 - With a little help . . .
- 3 What is R?
 - R is many things to many people
 - Developer communities
- 4 The R Development Process
 - R itself
 - CRAN, Bioconductor, github, others
 - Opportunities and challenges

R-spatial

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

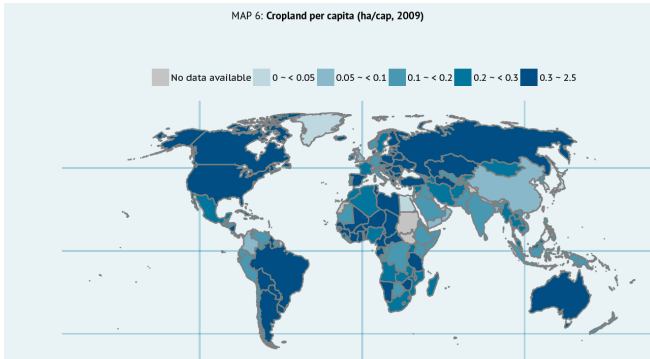
What does use of R-spatial look like?

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

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

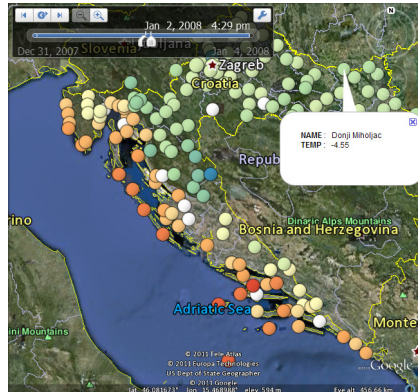
The 2013 FAO Statistical Yearbook

“The 2013 FAO Statistical Yearbook . . . has been created from beginning to end with the statistical software R and the typesetting language LaTeX: from data retrieval, to data processing, indicator construction, and blueprint-ready pdf file for distribution.”



Google Earth and **plotKML**

Partly to meet the needs of Global Soil Information Facilities, Tomislav Hengl has been coordinating activity on displaying output on Google Earth (admittedly not open source) in the **plotKML** package (http://gsif.isric.org/doku.php?id=wiki:tutorial_plotkml). Other packages use R graphical devices with contextual backgrounds from GE and OSM.



Mapping Hurricane Sandy damage

In a talk at useR! 2013, Charles DiMaggio presented a talk on using maps in disaster and emergency settings; he had with his students used maps made with R in deploying volunteers in the aftermath of Sandy. The talk is at http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/resources/sandy/useRslides.pdf.

Hurricane Sandy Housing Damage

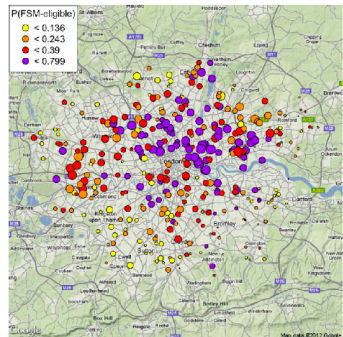


James Cheshire: `spatial.ly`

<http://spatial.ly/> is a successful and upbeat blog, evangelical in tone:

“The software has become established as one of the best around for statistics and it is becoming increasingly recognised as a tool for data visualisation and spatial analysis.”

(<http://spatial.ly/2013/04/analysis-visualisation-spatial-data/>)

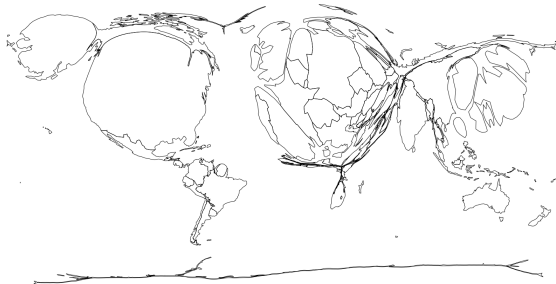


Ease of access and representativeness I

James Cheshire: “Thanks to the release of log files containing all hits to <http://cran.rstudio.com/> server it is possible to make a map showing the parts of the world with the most active R users (specifically those mostly using the RStudio interface).”

(http://spatial.ly/2013/06/r_activity/)

R Activity Around the World



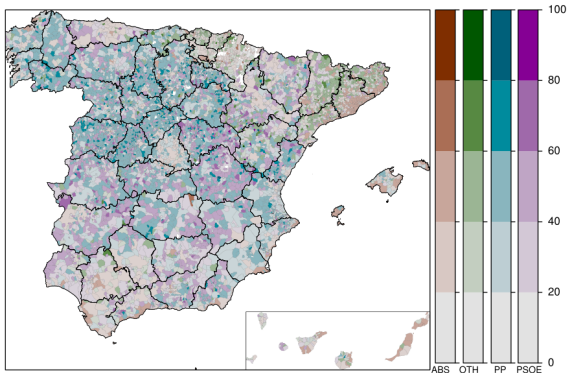
Ease of access and representativeness II

Oscar Perpiñán: “More or less explicitly they use the RStudio logs to give an answer to the question ‘How many people use this R package?’ In my opinion, such question cannot be answered safely with the RStudio download logs.

The RStudio mirror is a sample that cannot be safely regarded as representative of the mirrors network. The mirrors of the Comprehensive R Archive Network, mostly operated by public or nonprofit institutions, provide faster package downloads for users at their geographical location” (<http://procomun.wordpress.com/2013/06/15/rstudiologs/>).

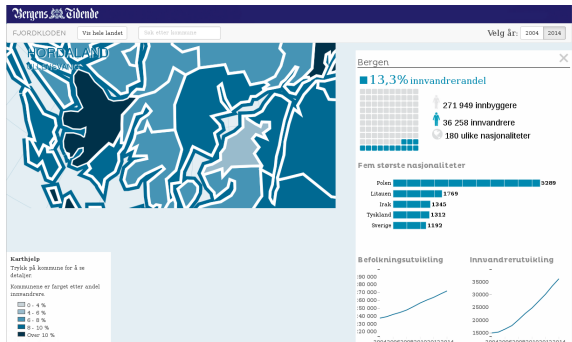
Displaying spatial data

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



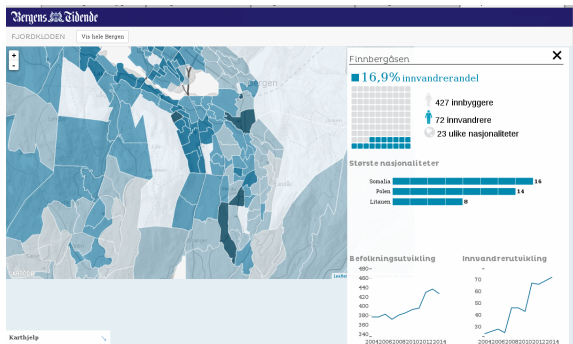
Bergens Tidende interactive graphics

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



Bergens Tidende interactive graphics

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



R spatial

- Albrecht Gebhardt did a lot of the early porting from S to R; there were also valuable contributions from spatial statisticians at Lancaster University, at UCAR, and many others
- Kurt Hornik, who runs CRAN, encouraged me to talk about R and GIS at the March 2001 Distributed Statistical Computing meeting in Vienna, at which I got to know active developers personally
- At a meeting in Santa Barbara in Spring 2002, I met other researchers who provided critical, occasionally very critical, comments and encouragement to continue fostering an R spatial community
- By the next DSC meeting in March 2003, I was organising a thematic session on spatial statistics, and a crucial fringe developers' workshop to discuss how to advance spatial data analysis in R

CRAN Spatial task view

Since 2003, a number of community-building steps have been made over and above developing contributed packages. From the CRAN side, the Spatial task view is the hub, to which traffic is channelled to package pages and to ancilliary websites, as well as the special interest group mailing list. There is also a task view for handling and analyzing spatio-temporal data

CRAN Task View: Analysis of Spatial Data

Maintainer: Roger Bivand
Contact: Roger.Bivand at nhh.no
Version: 2014-02-26

Base R includes many functions that can be used for reading, visualising, and analysing spatial data. The focus in this view is on 'geographical' spatial data, where observations can be identified with geographical locations, and where additional information about these locations may be retrieved if the location is recorded with care. Base R functions are complemented by contributed packages, some of which are on CRAN, and others are still in development. One active location is [R.Forge](#), which lists "Spatial Data and Statistics" projects in its [project tree](#). Information on R-spatial packages, especially [sp](#) will be posted on the R-Forge rspatial project [website](#), including a visualisation gallery.

The contributed packages address two broad areas: moving spatial data into and out of R, and analysing spatial data in R.

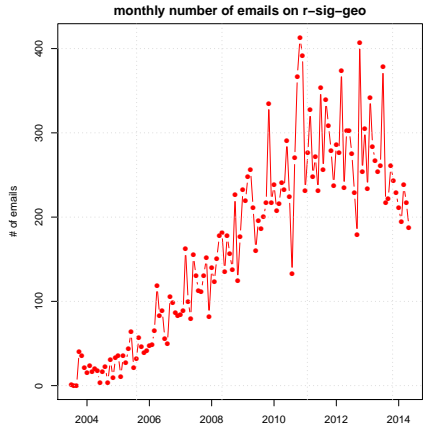
The [R-SIG-Geo](#) mailing-list is a good place to begin for obtaining help and discussing questions about both accessing data, and analysing it. The mailing list is a good place to search for information about relevant courses, and a list is hosted at the [GeoDaCenter](#).

The packages in this view can be roughly structured into the following topics. If you think that some package is missing from the list, please let me know. Please also visit and contribute to the [spatial data handling](#) and [spatial statistics](#) pages on the R Wiki.

- **Classes for spatial data** : Because many of the packages importing and using spatial data have had to include objects of storing data and functions for visualising it, an initiative is in progress to construct shared classes and plotting functions for spatial data. The [sp](#) package is discussed in a note in [R News](#). Many other packages have become dependent on these classes, including [rgdal](#) and [maptools](#). Functions provided by [vec2dtransf](#) for applying affine and similarity transformations on vector spatial data (sp objects). The [rgmisc](#) package provides an interface to topology functions for [sp](#) objects using [GEOS](#). [rgmisc](#) is now available for Mac OSX on CRAN. The [raster](#) package is a major extension of spatial data classes to virtualise access to large rasters, permitting large objects to be analysed, and

R-sig-geo mailing list

Following the 2003 workshop, we started a project on Sourceforge to permit joint development, and a mailing list served within the family of R lists from Zurich. Traffic on the list has grown steadily, with a subscribed membership in May 2014 of over 3100. Naturally, many of these “lurk” without posting, while others post without helping, and many fewer help by answering posted questions. This final group is however growing, and since the list archives are also kept on Nabble, they can be searched for information.



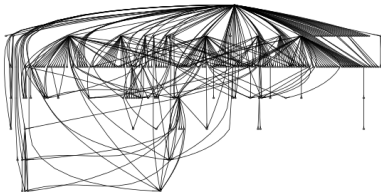
Spatial on R Forge

R Forge is used actively by individuals and groups in developing packages for spatial data analysis, with 100 projects registered in August 2012. Some projects are registered in more than one topical area, some may never mature, but some are already in active use; the **raster** package is already frequently discussed on R-sig-geo — it was released to CRAN in late March 2010 after a gestation of 16 months.

The screenshot shows the R-Forge website interface. At the top, there is a search bar and navigation links for Home, My Page, and Projects. Below the navigation is a 'Software Map' section with links to Tag cloud, Project Tree, and Project List. The main content area is titled 'Project tree' and shows a breadcrumb trail: 'Topic > Spatial Data & Statistics'. Under this, there are several project categories with their respective counts: 'Classes for Spatial Data (9 projects)', 'Disease Mapping and Areal Data Analysis (3 projects)', 'Ecological Analysis (48 projects)', 'Geostatistics (9 projects)', 'Point Pattern Analysis (8 projects)', and 'Reading and Writing Spatial Data (15 projects)'. To the right, there is a 'Browse By' section with a list of filters: 'Development Status', 'Environment', 'Intended Audience', 'License', 'Name', 'Natural Language', 'Operating System', and 'Programming Language'. Below the project list, there is a summary for 37 projects in the result set, displaying 20 per page, sorted by activity ranking. A specific project, 'Global Soil Information Facilities', is highlighted, with a description: 'Global Soil Information Facilities is an initiative of the ISRIC — World Soil Information Institute and Africa Soil Information Service (africansoils.net) team.' To the right of this project, there are statistics: 'Activity Percentile: 89.62', 'Activity Ranking: 12.00', and 'Registered: 2011-06-12 20:12'. At the bottom, there is a list of filters for this project: 'Development Status: 1 - Planning [Filter]', 'Environment: Console (Text Based) [Filter]', 'Intended Audience: Developers [Filter]', 'Intended Audience: End Users/Desktop [Filter]', 'License: OSI Approved: GNU General Public License (GPL) [Filter]', 'Natural Language: English [Filter]', 'Operating System: OS Independent [Filter]', 'Programming Language: Python [Filter]', 'Programming Language: R [Filter]', and 'Topic: Environmetrics [Filter]'.

The **sp** package

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



Spatial objects

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

Coordinate reference systems

- Coordinate reference systems (CRS) are at the heart of geodetics and cartography: how to represent a bumpy ellipsoid on the plane
- We can speak of geographical CRS expressed in degrees and associated with an ellipse, a prime meridian and a datum, and projected CRS expressed in a measure of length, and a chosen position on the earth, as well as the underlying ellipse, prime meridian and datum.
- Most countries have multiple CRS, and where they meet there is usually a big mess — this led to the collection by the European Petroleum Survey Group (EPSG, now Oil & Gas Producers (OGP) Surveying & Positioning Committee) of a geodetic parameter dataset

Coordinate reference systems

- The EPSG list among other sources is used in the workhorse PROJ.4 library, which as implemented by Frank Warmerdam, handles transformation of spatial positions between different CRS
- This library is interfaced with R in the **rgdal** package, and the CRS class is defined partly in **sp**, partly in **rgdal**
- A CRS object is defined as a character NA string or a valid PROJ.4 CRS definition
- The validity of the definition can only be checked if **rgdal** is loaded

Spatial*DataFrames

`Spatial*DataFrames` are constructed as Janus-like objects, looking to mapping and GIS people as “their” kind of object, as a collection of geometric features, with data associated with each feature. But to data analysts, the object “is” a data frame, because it behaves like one.



Spatial classes provided by **sp**

This table summarises the classes provided by **sp**, and shows how they build up to the objects of most practical use, the `Spatial*DataFrame` family objects:

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

Methods provided by **sp**

This table summarises the methods provided by **sp**:

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

Using `Spatial` family objects

- Very often, the user never has to manipulate `Spatial` family objects directly, as we have been doing here, because methods to create them from external data are also provided
- Because the `Spatial*DataFrame` family objects behave in most cases like data frames, most of what we are used to doing with standard data frames just works — like `[]` or `["$"]` (but no `merge`, etc., yet)
- These objects are very similar to typical representations of the same kinds of objects in geographical information systems, so they do not suit spatial data that is not geographical (like medical imaging) as such
- They provide a standard base for analysis packages on the one hand, and import and export of data on the other, as well as shared methods, like those for visualisation

Extension of `spatial` family objects

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

- **spacetime** provides classes and methods for spatio-temporal data, including space-time regular lattices, sparse lattices, irregular data, and trajectories
- **raster** is built around a number of S4 classes of which the `RasterLayer`, `RasterBrick`, and `RasterStack` classes are the most important; a notable feature of the **raster** package is that it can work with raster datasets that are stored on disk and are too large to be loaded into memory

PROJ.4

- The reliance of **sp** classes on the PROJ.4 string representation of coordinate reference systems was noted earlier
- XML-based alternatives could have been chosen, but PROJ.4 strings are also “human-readable”, and readily converted into other representations
- The version of the PROJ.4 library to which **rgdal** is linked is reported to make reproducible research somewhat easier; its behaviour needs careful tracking in a long-running application like R
- In the forthcoming version of the PROJ.4 library, it will be easier to check for metadata files, but it is still very difficult to access their versions, as we have seen recently

GDAL/OGR

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

GEOS

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

OSGeo GIS

- GRASS 6 was released in March 2005, and has now reached 6.4.3; the re-implemented interface package **spgrass6** works with GRASS 6 and 7
- **spgrass6** is loose-coupled, using GDAL on both sides of the interface to exchange vector and raster data by writing to and reading from a temporary directory
- From April 2009, **spgrass6** was revised to support a second mode of operation; the earlier way of using R within a GRASS session was supplemented by the ability to initiate a GRASS session from R, setting up the environment variables used by GRASS
- This was complemented by interfacing most GRASS commands directly in a cross-platform fashion, using the `-interface-description` flag that GRASS commands use to return their flags, parameters, etc.

What is R?

- 1 Introduction
 - Motivation: longer lives
- 2 R-spatial
 - What does use look like?
 - R-spatial — Building a community
 - R-spatial — **sp**
 - With a little help ...
- 3 What is R?
 - R is many things to many people
 - Developer communities
- 4 The R Development Process
 - R itself
 - CRAN, Bioconductor, github, others
 - Opportunities and challenges

R is many things to many people

- a software application for getting the work done (and/or for teaching)
- a statistical programming language with a community of users
- an ecology of contributed packages addressing the needs of different scientific communities
- communities of developers with different tastes, abilities and desires
- a value generating vehicle in the software, consulting and publishing businesses

R: a software application

- Well, we all use software applications — to “do stuff”; R began as a way of providing students with software to support statistics teaching
- Software applications are often provided by large organisations in standardised ways, deployed to many devices intended to have the same software
- Early R release dates were intended to match the university teaching year (but varying by hemisphere), so R was then seen as a software application
- Packages defined as *base* change in lock-step with R; most IT support providers only update between releases if there are security issues

R: a software application

- In addition to controlled software applications, we are now in a BYOD world with its challenges and opportunities
- Users, often with little insight into software installation or OS issues, expect “stuff” to work by magic on their computing platforms
- Communicating the advantages of the command line to such users is possible but not automatic, as scripting is not the first idea users associate with software applications
- This may explain some of the appeal of RStudio and similar script support (IDE) environments

R: a statistical programming language

- Twenty years ago, R was not only a software application for students of statistics, but also a project in the design of statistical programming languages (Ihaka and Gentleman, 1996; Gentleman and Ihaka, 2000; Ihaka and Temple Lang, 2008)
- Based on S, but crucially also on SICP (The Structure and Interpretation of Computer Programs), R has provided opportunities for further research and implementation in the design of statistical programming languages
- Internal aspects, such as byte compilation and the design of classes have been important
- A recent major advance is the introduction of reference classes in addition to old-style (S3) and new-style (S4) classes

R: an ecology of contributed packages (Mens et al., 2014)

- As with other language communities (Python, Perl, LaTeX), R's package mechanism and archive network has been a big success (Fox, 2009), now over 5000 packages on CRAN
- Theußl et al. (2011) and Ligges et al. (2010) point to challenges in running the archive network, including mutual dependencies between packages, and between packages and external libraries
- As Ligges et al. (2010) conclude, "being much stricter in CRAN maintenance," including the CRAN Repository Policy (CRAN Repository Maintainers, 2014), is unavoidable
- CRAN Task Views, coordinated by Achim Zeileis, are an essential part of the ecology, guiding rather than policing packages by topic

R: communities of developers

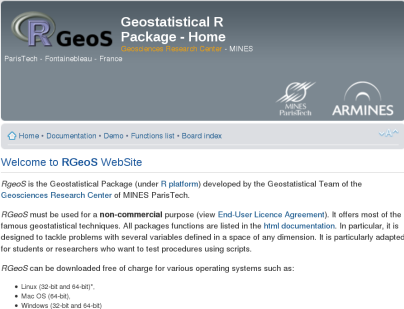
- With almost 1750 projects and over 8000 registered users, R-Forge has provided a flexible mechanism for collaborative package development (Theußl and Zeileis, 2009)
- Searching github for R language tags gives 8,426 repository hits, only some of which are packages, and many of which are single author; some are write once, read never
- While many R packages can be installed without compilation, Eddelbuettel (2013) shows how **Rcpp** makes C++ available in an accessible way; R 3.1 offers platform dependent C++11 support
- It is positive that developer tastes vary, as multiple implementations using different technologies offer ecological benefits (provided their outcomes are compared)

R: a value generating vehicle

- While the value generated by R has always been substantial, it was until recently in the form of better teaching and research than would have been achieved at the same cost
- Value generated by R is captured within organisations using R for their research and production; many contribute back via the Foundation and conference sponsorship
- In addition, consultancies have come into being providing R customisation services, and providing R support to other businesses
- The concepts of citizenship and voice in free software communities are complex, with the risk that value generation may not perceive the needs that should be prioritised (mismatch of cultures)

Understanding citizenship . . .

The Geostatistical R package from Mines ParisTech in Fontainebleau is an example of the opportunities that arise at the junction of non-free software, commercial and grant-based research, and R. Their package has had to be distributed from their site in binary form, but the team is actively examining ways of releasing a re-named package on CRAN, to access the potential benefits to themselves of participating in a community, and to better accommodate reproducible research.



The screenshot shows the homepage of the RGeoS package. At the top, there is a dark blue header with the RGeoS logo (a stylized 'R' with a globe) and the text "Geostatistical R Package - Home". Below this, it says "Geosciences Research Center - MINES ParisTech - Fontainebleau - France". There are also logos for "MINES ParisTech" and "ARMINES". A navigation bar contains links for "Home", "Documentation", "Demo", "Functions list", and "Board index". Below the header, the main content area starts with "Welcome to RGeoS WebSite". A paragraph explains that RGeoS is developed by the Geostatistical Team of the Geosciences Research Center of MINES ParisTech. It states that RGeoS must be used for a non-commercial purpose and offers most of the famous geostatistical techniques. A list of operating systems is provided: Linux (32-bit and 64-bit), Mac OS (64-bit), and Windows (32-bit and 64-bit).

Root communities

- Starting from S, John Chambers and the Bell Labs “little languages” environment, simplicity and modularity are preferred
- Underlying coding in C and Fortran, often building on Statlib and Netlib resources, for example for linear algebra
- Clear links to functional programming rather than procedural programming (which matches Statlib/Netlib code); Luke Tierney’s background in XLispStat is important
- Closed list of developers with commit rights to the source code repository, private R-core mailing list archive for internal discussions and coordination

Extensions to root communities

- There are few movements in base or recommended packages, but additions occur from time to time, such as **compiler** (Luke Tierney), **parallel** (Tierney, 2008), and **Matrix** (Douglas Bates and Martin Maechler)
- Handling encoding and multibyte characters, as well as the translation of messages of various kinds (**translations**), attempt to help in the use of R outside English language locales
- Windows binaries are managed by Duncan Murdoch, as are tools to build R and R packages, including 64-bit versions
- OSX binaries are similarly managed by Simon Urbanek, where OS version road bumps cause much more work — Mavericks involved a lot of pre-planning (transition to Clang and LLVM compilers; see post by Brian Ripley 22 April on R-SIG-Mac)

Mailing lists, etc.

- The mailing lists have “always” been the main channel of communication between community members, where authoritative answers from core developers may be posted
- However, between 2010 and today, monthly traffic has dropped from over 5000 messages to about 2500, the same level as 2005
- Vasilescu et al. (2014) point to alternative channels (StackOverflow), but these are arguably dysfunctional, because often the lists are where bugs get reported, rather than through external fora or bugzilla
- Many books introducing and applying R have been published in the last five years; many blogs have appeared too, but it is hard to assess the authority for views advanced

Alternatives

- Does authority matter — well, it may not, but how one advances points of view does matter, perhaps especially checking before posting
- Recent questions on R-devel about C++11 led to an interesting discussion, and to the provision in 3.1 of suitable mechanisms on platforms with support
- It seems that there are broadly generational differences in the depth of checking expected, but these are also idiosyncratic
- The finding by Vasilescu et al. (2014) that reputation and track record matter also applies on the mailing lists, but explicit socio-technical incentives are not used

Alternatives

- Some alternatives are directed to providing other ways of working with R, others to replacing R
- RStudio and associated projects appear to take positions on how working with R might be advanced, but differ from R itself in what may be termed “cool”
- Radford Neal’s pqR is a drop-in replacement for R, based on R, but with speed improvements; some of these have been adopted in R itself
- Julia has been contrasted with R on fast development v. fast execution; “the technical programming language of the future” (Douglas Bates 2013)

The R Development Process

- 1 Introduction
 - Motivation: longer lives
- 2 R-spatial
 - What does use look like?
 - R-spatial — Building a community
 - R-spatial — **sp**
 - With a little help ...
- 3 What is R?
 - R is many things to many people
 - Developer communities
- 4 The R Development Process
 - R itself
 - CRAN, Bioconductor, github, others
 - Opportunities and challenges

The R Development Process

- As documented in “Software Development Life Cycle” (The R Foundation for Statistical Computing, 2014), the R development process can be defined for R itself, base packages released for each release of R, and recommended packages released independently of R releases
- The document describes R’s licence, software development and testing methodologies, and communications within R-core
- It also details the release cycle (yearly in April), reporting mechanisms for issues and bugs, and documentation
- An unstated assumption is that advanced users test pre-release betas and release candidates; this did not hold for 3.1.0 with regard to lost precision in reading doubles

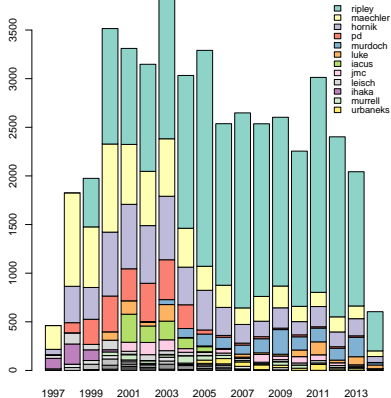
The R Development Process: sidelights

- Sometimes, sidelights are available for parts of the development process, for example Windows 64-bit support
- Section 3.1.10 of the R Installation and Administration manual says: “Support for MinGW-w64 was developed in the R sources over the period 2008–10 and was first released as part of R 2.11.0. 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.”
- This suggests active collaboration between R-core members and developers of upstream build train components; the R development process interacts with processes in other open source projects on which we depend

R itself

- The Developer Page is not much help with regard to knowing what R-core is thinking, but reading R-Devel is often very informative, sometimes provided that one reads “between the lines”
- It is very helpful to try to construct a picture of the priorities of R-core developers; arguably stability and backwards-compatibility have been given high priority
- In addition, stability is conditioned by forward-looking testing on a wide range of OS and platform variants
- The development version (R-devel, the trunk) is worth getting to know; tracking changes by RSS is very valuable (link from the Developer Page)

Commits to R trunk



The point made by Fox (2009) remains valid five years later, that about half of all commits to the R trunk have been made by one R-core member. The skill set needed to contribute, for example via patches, is specific, but within these limitations, well-motivated and tested non-core patches are accommodated.

Recent changes

- There are a number of examples of changes and extensions that have come into R recently, here are some:
- Reference classes have been discussed over a longer period, permitting passing by reference rather than by value, and allowing closer interfacing with passing by reference languages (via **Rcpp** for example)
- The coming of LLVM and Clang proved disruptive but with positive contributions in promoting cleaner code, as did the GCC Address Sanitizer
- C++11 support was discussed a good deal on the R-devel list, and arrived with R 3.1; however cross-platform support for C++11 is going to remain poor for some time to come (Martyn Plummer on R-devel, 10 April)

CRAN

- CRAN, the Comprehensive R Archive Network, has roots in CPAN (Perl) and CTAN (TeX), and makes contributed packages available for users to install
- These contributed packages (now > 5000) and their tests and examples constitute a revealed expression of how R is seen from outside R-core
- So while package authors and maintainers benefit from running `R CMD check -as-cran` on their source package tarball to find inconsistencies, there is a bigger picture
- The bigger picture, and a crucial part of the community, is the regular checking of the development version of R against the CRAN packages to see whether changes in R break packages

Why CRAN repository policy is important

- There are several reasons why CRAN policies are important, but the salient reason is the limited time that CRAN maintainers have to check packages
- If we value the existence of CRAN, we should follow the policies; if you want to discuss them, understand that discussion takes time from running CRAN
- Some policies, such as clear licence definitions, are general across Free Software archives, and affect larger user organisations' own policies
- “Please ensure that `R CMD check -as-cran` has been run on the tarball to be uploaded before submission. This should be done with the current version of R-devel”
- Using the development version to check is not scary — just use the Win-Builder site, which supports such checking

Criticisms of CRAN

- Because of the large number of packages, the Task View mechanism within CRAN and administered by Achim Zeileis is a key resource, and should be fostered
- It has been pointed out that the appearance of the CRAN website and the Task Views is very old-fashioned, but that does not mean that it is hard to find information
- CRAN has recently been contrasted with Bioconductor with respect to versioning; CRAN packages are not in lockstep with R releases, possibly making reproducible research more complicated
- In fact, judicious use of `sessionInfo()`, listing R and package versions and platform details, should be sufficient, unless there are external dependencies varying by version

Alternatives

- Bioconductor has a clearer policy with regard to package releases and documentation, perhaps more constraining than CRAN
- R-Forge — within the CRAN system — and RForge outside it provide collaborative development platforms for R packages; R-Forge also permits installation of development versions of packages
- More modern developers may prefer github, which is not run on servers made available to the R community by universities, and does not as such provide package checking or building support
- Only code in packages on CRAN is used to check the R development version, so changes in R can break other code, for example on github or R-Forge, silently

R-spatial: where from here?

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

R and CRAN

- The very specific and uncodified skill set in R development and maintenance, covering not only a mix of programming languages, but also attitudes to checking the consequences of commits
- A very uncommon mix of coding skills and judgement when working with and recalling multifaceted issues that can be hard to classify in a bug-tracker, blending compiler and numerical issues
- Recruitment is problematic; doing R, CRAN or R-Forge maintenance does not bring tenure, and very often insufficient understanding for the dilemmas faced by maintainers is shown
- CRAN policy is often criticised on R-devel, but usually by posters who haven't tried to grasp the challenges faced by very few maintainers

R communities

- Quite a number of the challenges for communities come from CS ontology dissonances (intra-community chatter crowds out inter-community communication); that is, understanding citizenship inclusively
- It remains vital that everyone compares outcomes with different implementations (scripts, packages); if you did not try out an R beta, don't complain if your production gets broken at release
- CRAN checks package examples; we should aim to check as much published code as possible for continued executability (Tal Galili, R-bloggers aggregator)
- Apparent tendencies to pre-emptive assumptions (that we know best by definition) from RStudio/github "guys" are less than constructive, and a good deal of diplomacy is needed

Opportunities and alternatives

- A key opportunity has always been that R permits shared communication between and within disciplines, and helps foster appropriate statistical practice
- R is potentially at the forefront of reproducible research, data exchange, and responsibility in the face of the “boosting” of results (but can do GIGO too, of course)
- Constructive “voice” — ‘I think this needs improvement or replacement and here is my patch, which I have tested as well as I can’ — is always worthwhile
- R will benefit from the maturing of alternatives such as Julia, and movements between projects among developers should be seen as a useful sharing of experience

References

- CRAN Repository Maintainers (2014). *CRAN Repository Policy*.
- Eddelbuettel, D. (2013). *Seamless R and C++ Integration with Rcpp*. Use R! Springer, New York.
- Fox, J. (2009). Aspects of the Social Organization and Trajectory of the R Project . *The R Journal*, 1(2):5–13.
- Gentleman, R. and Ihaka, R. (2000). Lexical scope and statistical computing. *Journal of Computational and Graphical Statistics*, 9:491–508.
- Ihaka, R. and Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.
- Ihaka, R. and Temple Lang, D. (2008). Back to the future: Lisp as a base for a statistical computing system. In Brito, P., editor, *Proceedings in Computational Statistics*, COMPSTAT, pages 21–34, Heidelberg. Physica-Verlag.
- Ligges, U., Hornik, K., Murdoch, D., Ripley, B., and Theussl, S. (2010). *Prospects and challenges for CRAN*.
- Mens, T., Claes, M., Grosjean, P., and Serebrenik, A. (2014). Studying evolving software ecosystems based on ecological models. In Mens, T., Serebrenik, A., and Cleve, A., editors, *Evolving Software Systems*, pages 297–326, Berlin. Springer.
- The R Foundation for Statistical Computing (2014). *R: Software Development Life Cycle: A Description of R's Development, Testing, Release and Maintenance Processes*.
- Theußl, S., Ligges, U., and Hornik, K. (2011). Prospects and challenges in r package development. *Computational Statistics*, 26:395–404.
- Theußl, S. and Zeileis, A. (2009). Collaborative Software Development Using R-Forge . *The R Journal*, 1(1):9–14.
- Tierney, L. (2008). Implicit and explicit parallel computing in R. In Brito, P., editor, *Proceedings in Computational Statistics*, COMPSTAT, pages 43–54, Heidelberg. Physica-Verlag.
- Vasilescu, B., Serebrenik, A., Devanbu, P., and Filkov, V. (2014). How social q&a sites are changing knowledge sharing in open source software communities. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & #38; Social Computing*, CSCW '14, pages 342–354, New York, NY, USA. ACM.