

Reproducible Cartography

Timothée Giraud and Nicolas Lambert

Abstract This paper deals with the production of statistical maps as part of the wider reproducible research paradigm. The current and most widespread ways to produce statistical maps combine several software products in a complex toolchain that use a range of data and file formats. This software and diversity of formats makes it difficult to reproduce the same analysis and maps. The aim of this paper is to put forward a unified workflow that allows map production in a reproducible process. We suggest hereby the `cartography` package, an extension of the R software, that fills the need of specific thematic mapping solutions.

Keywords Reproducibility · Open-source · R · Statistical cartography · Map workflow

1 Introduction

Scientific claims have to be supported by evidence. The assessment of scientific results is possible through the availability of methods and data used by scientists, and reproducibility is a key element to validate studies. The main idea behind reproducible research is to release studies along with data and computer code that support scientific findings (Peng 2011).

This idea has been first addressed by Jon Claerbout, a geophysicist who established a standard toolchain to produce (and reproduced) each figure made in his department (Claerbout and Karrenbach 1992). The will to establish standards or to develop reproducibility is a key point in the reproducible research movement

T. Giraud (✉) · N. Lambert (✉)
Unité Mixte de Service RIATE - Centre National de La Recherche Scientifique,
Paris, France
e-mail: timothee.giraud@ums-riate.fr

N. Lambert
e-mail: nicolas.lambert@ums-riate.fr

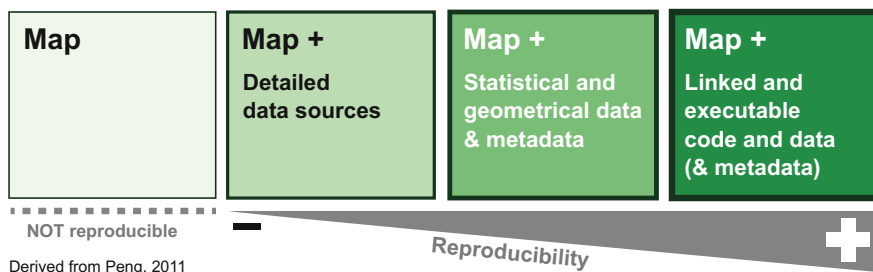


Fig. 1 The spectrum of map reproducibility

(Stodden and Miguez 2013) and scientific journals are increasingly keen on including datasets and code along with published articles (Peng 2009).

Discussions about reproducible research mostly take place in computational or statistical science fields. We argue that maps, as other graphics or statistical outputs, are part of scientific studies and should, as much as possible, be reproducible.

To be considered as evidence, a map should be open to debate and its construction should be transparent. Yet, maps produced in an academic context are currently made with a set of software products (spreadsheet, statistical software, GIS...) that slices the cartographic process. This multiplicity of tools and formats is an impediment to reproducibility. A fully reproducible map should be associated with the code and data used to produce it. Figure 1 describes what could be the spectrum of reproducibility applied to maps.

The first section of the paper demonstrates the relevancy of programming language solutions in mapping processes, especially in scientific contexts. In the second section, we put forward a unified workflow that allows map production in a reproducible process. We suggest hereby the `cartography` package, an extension of the R software that fills the need of specific thematic mapping solutions.

2 From GUI to Script

2.1 A Step Backward?

Most maps produced in an academic context are currently made with GIS or mapping applications that use graphical human-computer interactions. The use of graphical user interface (GUI) in computer science in general and cartography in particular is irrefutably a step toward more user-friendliness. But, this step may lead to the impossibility of reproducibility. To circumvent this weak reproducibility ability, most applications provide languages to build maps in their framework (e.g.,

Model Builder for ArcGIS or Python to create scripts in QGIS). However, these solutions hardly cover the full workflow from raw data to graphic representation and statistical findings.

Programming languages explicitly build on the idea of keeping traces of computations (data management, statistics, graphics and hence maps). Moreover, three main advantages to advocate the use of these scripting solutions are:

- (1) The possibility of combining statistical operations before and after spatial operations within the same tool (i.e., unified toolchain). Building a map implies to implement a set of steps: data management, statistical analyses, geo-processing operations and eventually graphical display of results. Using two, three or more software products to conduct these operations introduces ruptures and a multiplication of data and file formats.
- (2) The ability of using literate programming reports (i.e., reproducibility). Literate programming, introduced by Knuth (1992), strongly links analyses, statistical outputs and the code used to produce them in a single document. For example, R Markdown and Jupyter notebooks are respectively R and Python literate programming solutions based on the Markdown markup language.
- (3) Most of scripting languages are open-source and the use of open-source frameworks provides full transparency on the methods and tools used to conduct analyses.

Scripting solutions can appear as a step back for cartographers who learned map making with languages such as ARC Macro Language (introduced by ESRI in 1986 or SAS macros. We argue that in the reproducible research framework, researchers must use literate programming solutions that enable the full traceability of their studies. A script that describes every step of the process, from raw data to comprehensive vector thematic maps that includes all statistical and geometrical transformations, is considered a full metadata document.

2.2 R as a Go-To Tool for Integrated Analysis

Several programming languages and software solutions are used to carry out studies in a unified workflow that integrates data handling and data processing (including spatial analyses procedures and spatial representations). Python and R emerge as the most prominent.

Python appears to be more versatile than R which is more focused on data. Spatial libraries exist for these two languages but the R spatial ecosystem is more dynamic and includes more spatial analyses methods. As statistical cartography practitioners, we have decided to invest in R development that allows a strong connection between statistics and cartography.

R is both a language and an environment for statistical computing and graphics. A large part of this open-source software popularity is due to its plug-ins, called packages. R packages are created by users and distributed through public repositories (Comprehensive R Archive Network, commonly referred to as CRAN). As of September 2016 there were 9387 packages. This system allows the development and sharing of software pieces in a transparent way.

Among its many packages, some are dedicated to the management, modification and display of spatial features. Three main packages are unavoidable: `sp` (Pebesma and Bivand 2005; Bivand et al. 2013), `rgdal` (Bivand et al. 2016) and `rgeos` (Bivand and Rundel 2016). `sp` is used to manage spatial data (vector and raster). To manipulate geographical projections and data import/export, `rgdal` provides bindings to the GDAL (Geospatial Data Abstraction Library, a translator library for raster and vector geospatial data formats) and PROJ (Cartographic Projections Library). `rgeos` is an interface to GEOS (Geometry Engine, Open Source, a geometrical operation library) aiming at geo-processing spatial features.

Several packages already allow to produce thematic maps in R. Most of them are described in the *Visualisation* topic of the “Analysis of Spatial Data” CRAN Task View (CRAN Task Views are guides that group and describe packages of specific domains). Nevertheless, none of them fits the need to combine easy mapping features with an elegant design of functions, a complete set of cartographic layout elements (arrow, scale, legends...) and a large set of representations.

With the `cartography` package, we offer an extension of the R ecosystem that enables fully reproducible cartography linked to data collection and data analysis. `cartography` is built upon and benefits from two of the main related spatial packages (`sp` and `rgeos`).

3 The Cartography Package

3.1 Design

The aim of `cartography` is to design high quality thematic maps similar to those designed with common mapping and GIS solutions. `cartography` targets two categories of users: cartographers new to R programming or R users new to cartography. Therefore, its functions have to be easy to use for cartographers and ensure compatibility with standard R workflows.

The package design follows some of the current practices in GIS workflows. Each function has two main arguments: a spatial object (e.g., GIS: vector layer/R: `sp` object) and an attribute table (e.g., GIS: csv/R: data frame). Each function

focuses on a single cartographic representation (e.g., proportional symbols or choropleth representation) displayed on a georeferenced plot. This solution handles each representation as a layer and allows overlaying multiple representations on a map.

Functions have many parameters to fine-tune the cartographic representation. These parameters are similar to those used in GIS environments (e.g., width of the polygons boundaries, color palettes used in choropleth maps, customizable legend). Some additional functions are dedicated to layout design: customizable scale bar, a north arrow, a title, sources, and author information.

Following the common practice in R package development, each function in `cartography` is illustrated by a full example. Examples use datasets provided in the package (spatial and statistical datasets on European regions) and new users can easily reproduce them or adapt these scripts to their own data. A large range of export formats are available, in both vector and raster files, allowing final edition a graphic design software if needed (Giraud and Lambert 2016).

3.2 *Main Features*

`cartography` contains 35 functions that process and display spatial objects or map layout elements. Figure 2 gives an overview of the main features available in the package, indicating which functions are involved. Classical representations and some more advanced maps (that usually need geo-processing) are listed. The following figures are examples of maps solely built with `cartography`. The scripts used to generate them are available in a public repository (<https://github.com/riatelab/ReproducibleCartography>).

Figure 3 shows four maps produced with `cartography`. The first one uses the proportional symbols function (`propSymbolsLayer`) and displays the amount of population by country in Europe. The second one is a choropleth map (`choroLayer`) of the gross domestic product per inhabitant in euros.

The third and fourth maps combine two indicators. Symbol sizes are proportional to the amount of population and color scales are based on the distribution of gross domestic product per inhabitant (`propSymbolsChoroLayer`, `propSymbolsLayer + choroLayer`).

The three next maps are more complex. Their production involves geo-processing operations, usually scattered across multiple steps and intermediate files. `cartography` simplifies the process by wrapping all operations in its functions.

The “grid-cell” representation is a method that overcomes the arbitrariness and irregularity of administrative divisions. It highlights the main spatial trends by splitting the territory in regular blocks. Statistical values are distributed over a

Symbolgy	Characteristics	Function	Geometries		Visual Variables	Overview
			inputs	output		
Choropleth layer	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	choroLayer	▲ → ▲		Value, intensity	
Typology Layer	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	typoLayer	▲ → ▲		Hue	
Proportional Symbols Layer	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	propSymbolsLayer	▲ → ●		Size	
Colorized Proportional Symbols Layer (relative data)	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	propSymbolsChoroLayer	▲ → ●		Size, value, intensity	
Colorized Proportional Symbols Layer (qualitative data)	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	propSymbolsTypoLayer	▲ → ●		Size, hue	
Double Prop Symbols Layer	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	propTrianglesLayer	▲ → ●		Size, hue	
Regular grids	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	getGridData getGridLayer choroLayer	▲ → ■		Value, intensity	
«Points Bertin»	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	getGridData getGridLayer propSymbolsLayer	▲ → ●		Size	
Dot Density Layer	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	dotDensityLayer	▲ → ●			
Isopleth Layer	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	smoothLayer choroLayer	▲ → ●		Value, intensity	
Discontinuities	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	disclLayer getBorders getOuterBorders	▲ → √		Size	
Prop flows	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	GetLinkLayer propLinkLayer	▲ → √		Size	
Qualitative flows	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	GetLinkLayer gradLinkTypoLayer	▲ → √		Hue, size (classes)	
Graduated flows	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	GetLinkLayer gradLinkLayer	▲ → √		Size (classes)	
Layout	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	layoutLayer	▲ → □ EXTENT			MAP TITLE
Labels	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	labellayer	▲ → ●			bbb ccc AAA ddd eee
OSM basemap	☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷	tilesLayer	▲ → □ EXTENT			

☰ Associative ☱ Selective ☲ Order ☳ Quantitative

Fig. 2 Main features of cartography

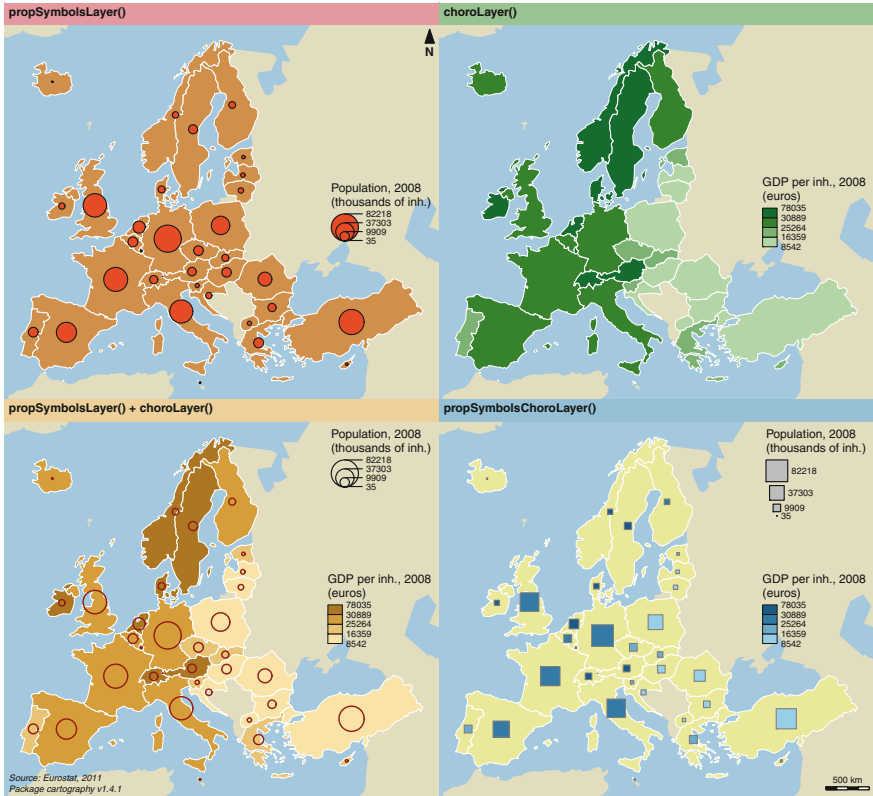


Fig. 3 Four classical maps

regular grid. Cell values are calculated taking into account the share of intersected areas between initial units and grid cells, and are eventually discretized and the grid is displayed as a choropleth map.

In cartography, this process relies on three functions. `getGridLayer` builds a regular grid (squares or hexagons) based on a spatial object and provides the shares of intersected areas. `getGridData` computes data that matches the grid layer. Finally, `choroLayer` displays the grid on a choropleth map (Fig. 4).

Discontinuity maps are based on the variation of a phenomenon in contiguous administrative units. This kind of representation does not focus on homogeneous regions, but rather on spatial breaks. Here, the thickness of the borders points out discontinuities intensities.

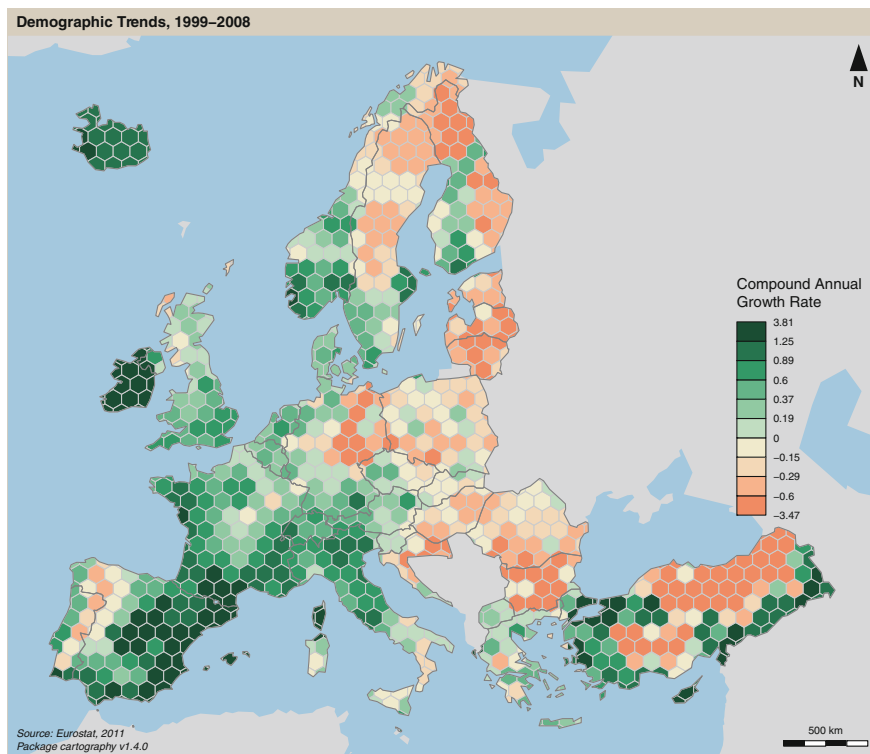


Fig. 4 Gridded map

The first step to build these maps is to extract borders from contiguous spatial units. The second step is to compute a discontinuity measure (either a ratio or an absolute difference). The third step consists in applying various values to the thickness of borders. Combining these discontinuities with a choropleth representation helps understanding the discontinuity direction (which one of two regions records the higher value).

In cartography, `getBorders` is used to extract borders between spatial units. `getOuterBorders` function can compute borders between non-contiguous units (maritime borders). `discLayer` function computes and displays discontinuities. A line's width reflects the discontinuity level between two neighboring units. The discontinuities layer can be supplemented by a choropleth layer (`choroLayer`). The resultant discontinuities representation is shown in Fig. 5.

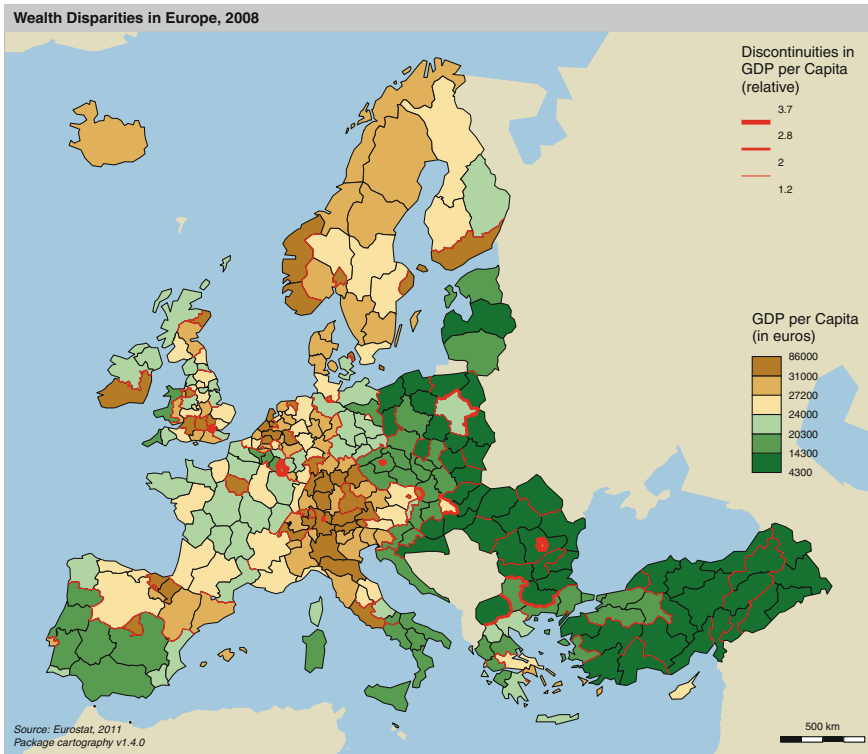


Fig. 5 Discontinuities map

Unlike choropleth maps that imply a phenomenon is spatially discrete, isopleth maps are based on the assumption that a phenomenon is spatially continuous. These maps use a spatial interaction modeling approach aiming at computing indicators based on stock values weighted by distances (Stewart 1942). This spatial representation is independent from the initial heterogeneity of the territorial division. The result is easy to read and can be considered as a bypass of the Modifiable Areal Unit Problem (MAUP) (Openshaw and Taylor 1979).

In cartography the `smoothLayer` function, that heavily depends on `SpatialPosition` (Giraud and Commenges 2016), allows a quick computation of these maps. The function takes a marked point layer and a set of parameters (a spatial interaction function and its parameters) as inputs and displays an isopleth map layer (Fig. 6).

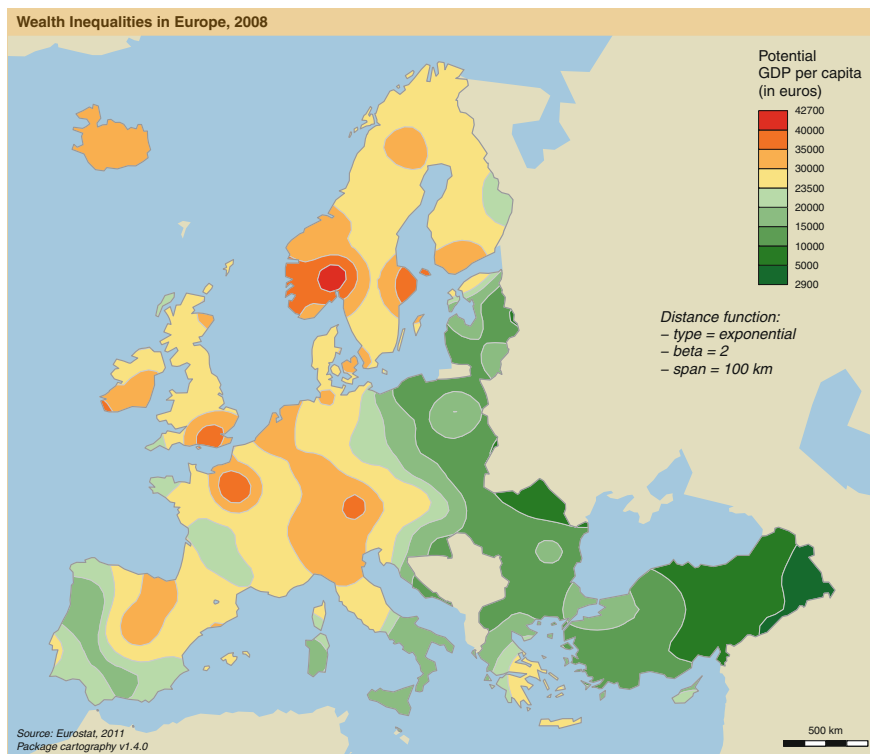


Fig. 6 Isopleth map

4 Conclusion

cartography was uploaded to CRAN in October 2015 and has been updated four times since (current version is 1.4.1). The development of the package happens on the online version control system GitHub (<https://github.com/Groupe-ElementR/cartography>). As of February 2017, the package has been downloaded 10,500 times from the RStudio CRAN. This number is an underestimate because CRAN is deployed across various mirror sites and only the RStudio site gathers this information. The number of downloads reveals an actual need of thematic mapping tools in reproducible research workflows.

Finally, we argue that maps should be more reproducible in order to obtain full acknowledgement as scientific productions. R and cartography (among other possibilities) could be relevant tools to ensure reproducibility in the academic context, hence promoting transparent scientific debates.

References

- Bivand, R., Keitt, T., & Rowlingson, B. (2016). *rgdal: Bindings for the geospatial data abstraction library*. <https://CRAN.R-project.org/package=rgdal>.
- Bivand, R., & Rundel, C. (2016). *rgeos: Interface to geometry engine—Open source (GEOS)*. <https://CRAN.R-project.org/package=rgeos>.
- Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied spatial data analysis with R* (2nd ed.). Springer, NY. <http://www.asdar-book.org/>.
- Claerbout, J., & Karrenbach, M. (1992). Electronic documents give reproducible research a new meaning. In *Proceedings of 62nd Annual International Meeting of the Society of Exploration Geophysics* (pp. 601–604).
- Giraud, T., & Commenges, H. (2016). *SpatialPosition: Spatial position models*. <https://CRAN.R-project.org/package=SpatialPosition>.
- Giraud, T., & Lambert, N. (2016). Cartography: Create and Integrate Maps in your R Workflow. *JOSS*, 1(4). doi:10.21105/joss.00054.
- Knuth, D. E. (1992). Literate programming. *CSLI lecture notes, Stanford, CA: Center for the study of language and information (CSLI)*, 1992, 1.
- Openshaw, S., & Taylor, P. J. (1979). A million or so correlation coefficients: three experiments on the modifiable areal unit problem. *Statistical Applications in the Spatial Sciences*, 21, 127–144.
- Pebesma, E. J., & Bivand, R. S. (2005). Classes and methods for spatial data in R. *R News*, 5(2), 9–13.
- Peng, R. D. (2009). Reproducible research and biostatistics. *Biostatistics*, 10(3), 405–408.
- Peng, R. D. (2011). Reproducible research in computational science. *Science*, 334(6060), 1226–1227.
- Stewart, J. Q. (1942). A measure of the influence of a population at a distance. *Sociometry*, 5(1), 63–71.
- Stodden, V., & Míguez, S. (2013). Best practices for computational science: Software infrastructure and environments for reproducible and extensible research. Available at SSRN 2322276.