2016

Building applications with FOSS4G bricks: two examples of the use of GRASS GIS modules as a high-level "language" for the analyses of continuous space data in economic geography

Moritz Lennert

Department of Geosciences, Environment and Society, Free University of Brussels (ULB)

Building applications with FOSS4G bricks: two examples of the use of GRASS GIS modules as a high-level "language" for the analyses of continuous space data in economic geography

Moritz Lennert [a, *]

[a] Department of Geosciences, Environment and Society, Free University of Brussels (ULB)
– moritz.lennert@ulb.ac.be

KEY WORDS: GRASS GIS, economic geography, spatial modeling, raster analysis, programming

ABSTRACT:

In a world where researchers are more and more confronted to large sets of micro- data, new algorithms are constantly developed that have to be translated into usable programs. Modular GIS toolkits such as GRASS GIS offer a middle way between low-level programming approaches and GUI-based desktop GIS. The modules can be seen as elements of a programming language which makes the implementation of algorithms for spatial analysis very easy for researchers. Using two examples of algorithms in economic geography, for estimating regional exports and for determining raster-object neighbourhood matrices, this paper shows how just a few module calls can replace more complicated low-level programs, as long as the re- searcher can change perspective from a pixel-by-pixel view to a map view of the problem at hand. Combining GRASS GIS with Python as general glue between modules also offers options for easy multi-processing, as well as supporting the increasingly loud call for open research, including open source computing tools in research.

---

1   Introduction

Science is constantly evolving and new indicators and analyses have to be developed. All to often, researchers reinvent the wheel by programming their algorithms in low-level code. This can possibly make this code very machine efficient if it is coded correctly, but it can also lead to a waste of time and energy that could better be spent on the content instead of the programming. Obviously many efforts to wrap existing, and well-tested, routines into libraries and other forms of packages exist, and most researchers use those. In spatial sciences, however, it seems that re- searchers often either use very high-level GUI-oriented desktop analysis systems, or they use fairly low-level systems, such as, for example, the fiona and rasterio Python libraries or their equivalent in other programming languages.

In the FOSS4G ecosystem, there is a middle species, though: tools that are composed of multiple command-line modules that can be chained to form very efficient work processes. In this approach each module can possibly be viewed as a very high-level programming function, but since they exist as independent executables, these modules can very easily be chained using standard shell scripting, thus easily building complex applications. Examples of such modular systems are OS- SIM, SAGA, the Orfeo Toolbox, to a certain extent the GDAL and OGR command line tools and others. One of the oldest of these toolboxes is GRASS GIS.

The use of such existing, and highly optimized, bricks, which, in this context, can almost be considered as the elements of a "programming language" of their own, allows to quite rapidly construct a program for calculating desired spatial indicators, generally using some general-purpose programming language as glue. One can, therefore, concentrate on the higher-level logic and not on the concrete implementation. The modular structure, with each module being its own executive, makes the programs chaining them particularly robust, with failure in one module not necessarily meaning failure in the rest. It also provides an easy entry point to speeding up analyses through high-level parallel processing. In addition, the fact that all the components are open source allows to possibly adapt them if necessary, but, more importantly in a research context, allows to control how they work.

In this article I, thus, argue that using existing FOSS4G modular software al- lows the researcher in economic geography to easily elaborate efficient analyses on large datasets without having to focus on every aspect of programming. I present two examples of algorithms useful in economic geography (and other spatial disciplines) that were easy to code using a modular system such as GRASS GIS: the main example concerns the elaboration of new indicators for estimating regional exports based on firm micro-data. A second, small, example covers calculating a neighborhood matrix for raster objects. I begin by presenting the tools more in detail. I then explain each of the two use cases and the way they were implemented in GRASS GIS.

2   The tools

2.1   A modular GIS for efficient raster processing and projection handling

GRASS GIS (Neteler et al., 2012) is a generalist GIS software package that has been under constant development for over 30 years. Although it now offers very sophisticated vector handling, at its origin it was almost purely raster based and as such its raster routines have known a constant scrutiny and improvement over a long period of time. Because of this history, with much of the code written in times of much lower machine performances, many of these routines are particularly memory efficient and can thus easily handle large data sets.

GRASS GIS can be used as a straightforward desktop GIS, but its aforementioned modular structure invites the user to construct targeted applications through chaining of module calls. Two elements make the use of a GIS toolbox such as GRASS GIS particularly interesting for economic geography, especially when working in continuous space: i) handling location, space and projections and ii) existing (efficient) matrix/raster processing routines.

By definition economic geography studies the spatial aspect of economic production and development. In that context the notions of location and distance take up particular importance. Many theoretical economic geography studies use abstract representations of space, often only including a virtual distance in a virtual plane, without real localisation. Empirical studies on real firms in real space, however, need to be localised using the relevant coordinate systems: either non- projected latitude-longitude coordinates in degrees on a given ellipsoid or

projected coordinates in meters on a plane defined by a specific projection system. Often different sources provide their data in different projection systems and data has to be reprojected from one to the other in order to be able to combine these different sources. Handling such projection issues is not easy, and errors can lead to wrong results, including erroneous distance measurements. As most of the FOSS4G ecosystem, GRASS GIS relies on a well-tested and constantly peer-reviewed open source system of projection handling, proj.4, and provides sophisticated options for projection management. Instead of having to worry about these issues on library level, the researcher can thus rely on these tools to provide the necessary processes with sufficient accuracy.

The second advantage of using GIS software when working in continuous space is the existence of routines handling raster data as a whole, including raster data that do not fit into the computer's memory. Using modules as bricks the researcher does not have to worry about the individual treatment of pixels, nor about processing data in chunks (row-tiles or tiles). GRASS GIS has particularly excellent memory handling, using something that can be considered as disk-based external memory. Implementing such routines from scratch is quite complicated and thus, using well-tested existing tools avoids reinventing the wheel.

2.2 The glue: Python

Although one can use any shell scripting language to elaborate processing chains by combining modules, GRASS GIS specifically proposes a simple python scripting API that allows to call the individual GRASS GIS modules from within a Python script. This allows combining the power of the GRASS GIS modules with the ease of an interpreted language such as Python for the necessary glue and house-keeping. Python is open-source, widely used and well-documented, notably in the geospatial community, and I will not present generalities here. Two Python libraries, however, were particularly useful in the context of this study: the GRASS Python scripting API and the multiprocessing module.

2.3 The GRASS Python scripting library

GRASS GIS provides two Python application programming interfaces (API) to its functionalities. A very simple Python scripting API which mainly aims at easing calls to GRASS GIS modules from Python code, and in addition adds a small series of specific convenience functions, and a more sophisticated, more low-level, API which provides access to internal C-function via ctypes, in addition to higher-level module access (Zambelli et al., 2013). Both allow to easily chaining GRASS GIS modules while profiting from a more sophisticated scripting language than those offered by standard command line shells.

In the examples here, I use the simple scripting API, essentially the run command(), read command(), parse command(), etc functions that allow to launch a GRASS GIS module and, if relevant, to capture its output. The core modules used in the central algorithms can all be called through run command(). Other smaller tasks, such as calculating the total sum all pixel values or collecting individual pixel values, can be done by calling read /parse command, loading the result into Python variables that can then be treated through Python tools.

For the scientific programmer this API is particularly easy to use as the translation from the command-line usage of the GRASS modules to the GRASS Python scripting API is very straightforward. A classic development flow would therefore start with interactive usage of GRASS GIS modules, possibly via the graphical user interface, to go on to the assembly of these modules into a simple shell script to then be finalised in a more complete python script (possibly a new GRASS module) using the API. Another alternative is to construct the script using the graphical modeller which provides the option to export the process as a Python script.

The accelerator: Simple parallel processing using Python's multiprocessing module

Researchers have to handle ever larger data sets. Very sophisticated environments and tools exist for high-performance computing dealing with really "big data". There are data sizes, however, which push the normal desktop computer to its limits, but for which the use of complex distributed file systems and cluster computing seem a bit overkill. Through its already mentioned memory handling, many of GRASS GIS' raster routines are mostly CPU-bound, not memory-bound. Using the increasing number of cores in standard desktop machines, can thus provide an easily accessible approach to significantly speed up processing time.

This is where the Python multiprocessing module comes in. It makes it easy to distribute the computation load across multiple nodes. It does not provide multithreading, but allows to launch parts of a Python script as multiple parallel processes. This allows to do high-level parallel processing, enough to gain significant calculation speed for medium to large datasets. The modular structure of toolboxes such as GRASS GIS, provide an excellent opportunity to divide tasks into separate processes. Thus, without any advanced programming

knowledge and using just a few simple calls to the Python module's functions, one can easily spread the workload amongst the increasing number of cores on the typical desktop machine, but also amongst the much larger number of cores available in university computing centers.

3   Example 1: Elaboration of new indicators for flexible-scale estimation of exports

3.1   Continuous space analysis in economic geography

Economic geography has long used, and still is using, aggregate data, mostly in administrative units. Such data is generally easily available and, through its aggregation, does not represent a heavy data load. Many indicators have been developed for such aggregated data and much useful information has been extracted from such indicators.

It has been known for a long time, however, that such aggregated data can cause a series of issues, non the least of which the modifiable areal unit problem (Briant et al., 2010; Gehlke and Biehl, 1934; Grasland and Madelin, 2006; Openshaw, 1983). Some voices plead for a renewed economic geography based on continuous-space analysis of data. Tobler and Grasland et al., as well as, and more specifically in the field of economic geography, Arbia, all propose the use of continuous space in order to transcend the issues raised by discrete boundaries, or the use of the flexibility of aggregation provided by point-located micro-data (Arbia, 2001a; Arbia et al., 2009, 2014a; Grasland et al., 2000; Tobler, 1979, 1989).

Micro-data forms the basis of such analyses and more and more analyses exist based on such data. As stated by Grasland and Madelin, "the access to individual data is of course the ideal situation, not because the individual level is in all cases the most appropriate one to observe or model a phenomenon, but mainly because it gives the choice to observe information at all possible levels and for all forms of spatial partitions" (Grasland and Madelin, 2006, p. 26). Examples of the use of such micro-data include birth, survival and growth analysis of firms (Arbia, 2001a; Arbia et al., 2014a, 2014b), clustering (Boix et al., 2011), the link between agglomeration and exports (Farole and Winkler, 2013; Koenig, 2009) and the heterogeneity literature (Bernard et al., 2012; Greenaway and Kneller, 2007; Melitz and Redding, 2015; Redding, 2011). However, dealing with such data creates new challenges for analysis, such as the development of new forms of analysis in continuous space, but also more practical challenge of dealing with geocoding the data and treating large amounts of data in an efficient manner. This is where the use of existing GIS tools can facilitate the work of the researcher.

3.2   Estimation of regional exports

Depending on the country, we have more or less precise data about international exports of firms, even at regional level. However, we know little about the pro- portion of the economic production that is exported to other regions in the same country.

Generally, estimates of the share of exported production are based on notions of concentration, most notably on location quotients, or derivatives thereof. Such approaches are often severely criticized (Isserman, 1980; Richardson, 1985), but are nevertheless subject to ongoing research and use (Billings and Johnson, 2012; Crawley et al., 2013; Strotebeck, 2010; Tian, 2013), because of their simplicity, but also because of a lack of credible and feasible alternatives. One of the biggest issues with concentration based analyses is the issue of the size of the space for which concentration is measured, often leading to the already mentioned modifiable areal unit problems (Arbia, 2001b; Tiebout, 1956).

In order to avoid such problems, in line with the literature analyzed above, I propose to work in continuous space using micro-data by adapting two existing indicators to the estimation of regional exports, based on existing work by Huff (1964) and by Marcon & Puech (2010). The details of these new indicators are explained elsewhere (Lennert, 2015), so I will only present them briefly and concentrate here on the practical implementation of their calculation.

3.3   Huff-like indicator

Huff (1964) defined his famous model in the context of market area and market share analysis. It relates the sales capacity of a given unit j (measured in number of employees) to the sales capacity of all sales units in the

reference space, weighted according to their distance to the point i in space considered. The result gives the probability of resident at point i to consume at sales unit j:

$$P_{ij} = \frac{A_j/D_{ij}^{\gamma}}{\sum_j \left(A_j/D_{ij}^{\gamma} \right)}$$

I propose to use the same equation, but for all production units of a given sector, not only for retail, following the already mentioned basic idea that the more the production of a sector is concentrated in comparison to the consuming population the more this production is exported from the producing regions to other regions in the country. Consuming population can be employment, resident population or income, total production or any combination of these in order to combine intermediate and final consumption. In addition, this consuming population can be corrected by the relative trade balance of the sector at national level in order to take into account the amount of production that is exported or imported. In the following, however, I will concentrate on the simple version of the indicator as the mentioned additions do not change the fundamental logic.

The following algorithm represents the practical translation of the indicator into program flow:

- For each economic sector (NACE):
    - Extract all vector points representing firms of that NACE
    - Get γ of NACE (see below for details)
    - For each firm:
        o For each pixel:
            - Calculate distance to that firm
            - Calculate distance-weighted attraction to firm
    - For each pixel calculate sum of attractions to all firms
    - For each firm:
        o For each pixel:
            - Calculate probability of that pixel to buy products of firm
            - Calculate population of that pixel that buys products of firm

In the absence of empirical data, the choice of per-NACE γ coefficients is somewhat arbitrary, even in the original Huff model. For this exercise I simply linearly scaled the γ per NACE between 0 (no influence of distance) and 2 (very local sales area) depending on the distance between all points (pixels) in the total covered space and production units of that NACE. See the annex for the detailed algorithm.

In order to then analyze regional exports, several options are now open from there. One can:

- Calculate for each point in space the proportion of its population catered to by a specific area in space
- Calculate total population that is supplied by firms of given spatial units, possibly comparing that population to the resident population of these spatial units to deduce exports (thus going back to spatial aggregation, but after having calculated the spatial concentration in continuous space).

3.4   The modified Marcon-Puech indicator

Marcon & Puech (2010) propose a function, called the M function, that can be considered as a local location coefficient in varying radius reference spaces. The authors use it to measure the average spatial concentration of industries in a given sector. The function is defined as follows:

$$M_s(r) = \frac{\sum_i \left(\frac{\sum_{j \neq i}(c_s(i,j,r) \cdot w_j)}{\sum_{j \neq i}(c(i,j,r) \cdot w_j)} \right)}{\sum_i((W_s - w_i)/(W - w_i))}$$

where i,j = firms, r = radius, s = sector, c = neighborhood function [0,1] within a given r, w = weight of firm (e.g. employment), W = total sum of weights.

One can either run this program with a different radius for each sector depend- ing on the general geography of its production units (similar to the γ coefficient in the Huff-model), or with a fixed radius for all sectors, possibly testing for the effect of distance by looping through a series of fixed radii around the production units and estimating trade balance or export employment rate by pixel for each radius.

In order to estimate exports, I rewrite their equation in line with Isserman's modification of the classic location quotient (Isserman, 1980):

$$X_{sr} = (L_{sr}/L_{sn} - L_r/L_n) \cdot L_{sn},$$

where $L_{sr} = \sum_j \left(c_s(i,j,r) \cdot w_j \right)$ and $L_{sn} = W_s$..

Again, one can refine this equation with the introduction of different measures of consumption and by taking into account the sectors relative international trade balance, but I will not do that here.

Translating this equation into algorithmic logic gives the following program flow:

- For each NACE:
    - Create a raster map with sum of employment per 1km pixel
    - Determine radius of reference space
    - For each pixel (and the respective reference space around the pixel):
        - Calculate total employment in that NACE within the reference space
        - Calculate total employment of all NACE combined within the reference space
        - Estimate employment trade balance within the reference space based on location quotient (in comparison to national rates)
        - Possibly: If trade balance is > 0: Calculate total export employ- ment of the pixel by multiplying export employment rate by the total employment in that pixel

3.5 Translating these equations into GRASS GIS "language"

This section presents the practical implementation of the above theoretical developments in GRASS GIS. For each indicator, I present the modules used in the core of the analysis. I do not present the complete program flow, including general input output handling and intermediate storage of results, but only the part which implements the fundamental elements. In order to ease readability, I also do not present the code in its Python API form, but as pure GRASS module calls. The entire source code of each example of these export estimation indicators as well as the code for a more sophisticated evolution of the Huff-like indicator into a doubly-constrained spatial interaction model can be found at https://github.com/mlennert/RegionalTradeModel/.

3.6 Huff-like indicator

For the Huff-like indicator, the first loop across all pixels to calculate the distance- weighted attraction to a given production unit (i.e. the nominator in equation 1) was easily calculated with fundamentally two module calls for each production unit:

```
r.grow.distance MapOfFirm output=distance
r.mapcalc "PerPixelAttractiveness = NbrEmployees / (distance^gamma)"
```

Here, `distance` is a map with for each pixel the distance to the production unit. `r.mapcalc` is a map calculator that applies the given formula pixel by pixel, with `NbrEmployees` and `gamma` available respectively for each production unit or for each NACE code.

In order to calculate the share of the population of a pixel that is catered to by a specific production unit, again only very few module calls are necessary:

```
r.series file=ListOfAttractivenessMaps method=sum output=TotalSum
For each production unit:
 r.mapcalc "PerPixelShare = PerPixelAttractiveness / TotalSum"
 r.mapcalc "PerPixelConsumingPopulation = PerPixelShare *
    PerPixelPopulation"
```

`r.series` creates a map with pixel-by-pixel aggregate statistics of all input maps which are given here through a text file `ListOfAttractivenessMaps` that contains a list of all relevant map names.

The rest of the code is mostly just logistics (extracting the point of location of each firm from a vector map, transforming to raster, correcting the distance measure for the pixel in which the firm is located, compiling the text file with the list of input maps, etc) or housekeeping (taking stock of the existing maps, erasing temporary maps, etc).

3.7   The modified Marcon-Puech indicator

As for the Huff-like indicator, the main core of the calculation of the Marcon- Puech-inspired indicator consists of only three module calls, based on existing per-NACE maps of total employment per pixel:

```
r.neighbors input=EmploymentPerPixelPerNACE method=sum nsize=radius \
  output=NaceEmploymentNeighborhood

r.neighbors input=TotalEmploymentPerPixel method=sum nsize=radius \
  output=TotalEmploymentNeighborhood

r.mapcalc "ExportEmploymentNeighborhood = \
  (NaceEmploymentNeighborhood / NaceEmploymentReference - \
  TotalEmploymentNeighborhood/TotalEmploymentReference) \
  NaceEmploymentReference"
```

The module `r.neighbors` provides a map with moving window stats for each pixel in the radius `nsize` expressed in number of pixels.

3.8   Parallelization of the processes

For both of the above routines, where each calculation has to be done NACE by NACE (and sometimes even firm by firm), parallel processing can significantly increase processing speed, thus allowing to upscale the analyses to even larger data sets. Parallel processes fetch the next non-treated NACE code in an input queue listing of all sectors to be treated, launch the necessary processing and either add alphanumeric results to a result queue, or just create new output maps. The fact that each module call can be launched as a separate process, as each GRASS GIS module is an autonomous executable, this very high-level approach is straighforward to implement, by adding just a few lines of code. As much documentation exists, I will not go further into details here. The code will be in the complementary material accompanying the final version of the paper.

4   Example 2: Calculating neighborhood matrices of raster objects

The notion of proximity is central in spatial analysis and very much so in economic geography. Tobler's first law of geography states that "everything is related to everything else, but near things are more related than distant things" (Tobler, 1970). Many models in geography are based upon this law and many tools exist to evaluate the importance of proximity, notably spatial autocorrelation (Geary, 1954; Moran, 1950). In many of these analyses, the notion of neighborhood plays a particularly important role. Tools already exist in FOSS4G environments for the elaboration of neighborhood matrices. In GRASS GIS, the addon module v.neighborhoodmatrix does just that for vector polygons, profiting of the topological nature of the internal vector format (Lennert and GRASS GIS Development Team, 2014). R's spdep package provides flexible options for analyzing neighborhood relations (Bivand et al., 2013; Bivand and Piras, 2015). However, none of these seem to take into account the situation where objects (i.e. clumps of pixels) are represented in raster maps. Such objects can obviously be vectorized and submitted to vector-based algorithms, but sometimes going through such an extra step can be computation- ally intensive. In the context of the elaboration of tools for object-based image analysis for GRASS GIS, the need thus arose for the elaboration of a tool that can provide a neighborhood matrix of raster objects. In this, very brief, section I show how this was implemented in the form of a new publicly available addon module r.neighborhoodmatrix (Lennert and GRASS GIS Development Team, 2016).

Raster objects are defined by clumps of adjacent pixels that have the same pixel value. This value is considered the object id. The algorithm to determine a neighborhood matrix based on such maps is quite simple:

- For each direction (4, 8 or even more):
  - For each pixel i:
    - Check if neighboring pixel value $V_j \; != V_i$
    - If true: add i, j to the list of neighbors

> ▪ If false: ignore

The practical implementation of this algorithm in GRASS GIS is again also very straightforward and consists in basically two module calls for each direction:

```
r.mapcalc "NeighborMap = if( \
  OriginalMap[NeighborhoodModifier] != OriginalMap, \
  OriginalMap[NeighborhoodModifier], \ null())"
r.stats -1n input=OriginalMap,NeighborMap
```

The `r.mapcalc` call has to be read as follows: if(expression, then, else). For a 4-direction analysis, `NeighborhoodModifier` is one of [0,-1], [0,1], [1,0], or [-1,0], representing respectively the left, right, upper and lower neighbor. Each pixel in `NeighborMap` thus contains the id of the neighbor if it is different, and a null value otherwise.

With the '1' flag set r.stats exports pixel values one by one and if more than one map is given as input (as is the case here) each output row contains the values of the equivalent pixels of all input maps. The 'n' flag instructs it to ignore null values, thus only taking into account those pixels that actually are on the border between two objects. The GRASS Python scripting API's `read_command()` function is used to capture the output of `r.stats` into a Python list which is filled direction by direction. As the result of these two module calls is pixel by pixel, and not object by object, as well as direction by direction, the list contains multiple entries of neighborhood pairs. A last step thus consists in uniquifying the list of neighborhood relations. This can be done by a simple Python call:

```
unique_neighbors = list(set(neighbors))
```

The rest of the code just handles general input and output. This second small example shows, again, how the use of a modular GIS tool- box such as GRASS GIS, allows to code algorithms in a very high-level language which takes away all of the hassle of handling the spatial data in detail. In order to accelerate the process, it is obviously possible to divide the per-direction r.mapcalc and r.stats calls amongst several computational cores. Again, this can be done very easily using Python's multiprocessing module.

5   Conclusion

The main aim of this paper was to show that using the modules of a modular GIS toolbox such as GRASS GIS (or others such as SAGA or Orfeo Toolbox for image processing) as bricks allows to very easily and quickly construct more elaborate spatial analyses. As mentioned, the modules can be envisioned as elements of a particular programming language. In combination with a more general-purpose language such as Python, very sophisticated applications can be built quite easily.

The main difficulty, in my experience, is to change the paradigm of thinking from an individual data point for data point (in this case pixel for pixel) perspective to a map perspective, trusting the GIS modules to handle the looping across all of the data. Once this change of perspective is realized, what used to seem complicated algorithms suddenly become quite easy to implement.

GRASS GIS' excellent memory handling in its raster modules, fruit of its long history and some significant revisions along the way, brings the additional added value of freeing the researcher of having to deal with the memory issues raised by large data sets. GRASS GIS modules can process gigabyte-heavy datasets while only using a few hundred megabytes of memory.

For CPU-bound applications, the combination of the modular approach, potentially treating each call to a GIS function as a separate process, allows the use of very simple parallelization strategies, such as Python's multiprocessing module. This approach cannot always rival with specifically programmed, targeted multi-threaded routines, but it can be a very good compromise between the need for upscaling and the limited time and energy available for programming.

Thus, as soon as the analysis the researcher aims to implement goes beyond the standard set of tools a normal spatial analysis or GIS package provides, combining modular elements makes application development easier, and possibly more efficient by using simple multi-processing. Obviously, one is not limited to only one set of modules, but can combine, in one single application, calls to the modules of different toolboxes.

This is also where the strength of FOSS4G comes in: although many proprietary tools also provide such modular access to functionality, each additional toolbox adds to the overall licencing costs and it is not always easy to combine tools from different vendors. The free and open nature of FOSS4G projects, combined with the

communication between developers, such as at FOSS4G conferences, creates an ecosystem where teams collaborate and thus ease the combination of different tools, without extra costs. In addition, the open source nature also provides the possibility to review the code, down to the very routines provided by the toolboxes, something that has gained in importance as efforts increase towards openness and reproductibility of research. As others have argued, peer review of code is just as important as peer review of methods and results (Barnes, 2010; Ince et al., 2012; Nosek et al., 2015).

6   References

Anselin, L., 2000. Part 2 The Link between GIS and spatial analysis. Journal of Geographical Systems 2, 11–15. doi:10.1007/s101090050023

Arbia, G., 2001a. Modelling the geography of economic activities on a continuous space. Papers in Regional Science 80, 411–424. doi:10.1111/j.1435-5597.2001.tb01211.x

Arbia, G., 2001b. The role of spatial effects in the empirical analysis of regional concentration. J Geograph Syst 3, 271–281. doi:10.1007/PL00011480

Arbia, G., Cella, P., Espa, G., Giuliani, D., 2014a. A micro spatial analysis of firm demography: the case of food stores in the area of Trento (Italy). Empirical Economics 1–15. doi:10.1007/s00181-014-0834-6

Arbia, G., Copetti, M., Diggle, P., 2009. Modelling Individual Behaviour of Firms in the Study of Spatial Concentration, in: Fratesi, D.U., Senn, P.L. (Eds.), Growth and Innovation of Competitive Regions, Advances in Spatial Science. Springer Berlin Heidelberg, pp. 297–327.

Arbia, G., Espa, G., Giuliani, D., Dickson, M.M., 2014b. Spatio-temporal clustering in the pharmaceutical and medical device manufacturing industry: A geographical micro-level analysis. Regional Science and Urban Economics 49, 298–304. doi:10.1016/j.regsciurbeco.2014.10.001

Barnes, N., 2010. Publish your computer code: it is good enough. Nature News 467, 753–753. doi:10.1038/467753a

Bernard, A.B., Jensen, J.B., Redding, S.J., Schott, P.K., 2012. The Empirics of Firm Heterogeneity and International Trade. Annual Review of Economics 4, 283–313. doi:10.1146/annurev-economics-080511-110928

Billings, S.B., Johnson, E.B., 2012. The location quotient as an estimator of industrial concentration. Regional Science and Urban Economics 42, 642–647. doi:10.1016/j.regsciurbeco.2012.03.003

Bivand, R., Hauke, J., Kossowski, T., 2013. Computing the Jacobian in Gaussian Spatial Autoregressive Models: An Illustrated Comparison of Available Methods: Computing the Jacobian in Spatial Autoregressive Models. Geographical Analysis 45, 150–179. doi:10.1111/gean.12008

Bivand, R., Piras, G., 2015. Comparing Implementations of Estimation Methods for Spatial Econometrics. Journal of Statistical Software 63. doi:10.18637/jss.v063.i18

Boix, R., Lazzeretti, L., Oliver, J.L.H. s, Molina, B.D.M., Ruiz, B.T., 2011. Creative clusters in Europe: a microdata approach (ERSA conference paper No. ersa11p471). European Regional Science Association.

Briant, A., Combes, P.-P., Lafourcade, M., 2010. Dots to boxes: Do the size and shape of spatial units jeopardize economic geography estimations? Journal of Urban Economics 67, 287–302. doi:10.1016/j.jue.2009.09.014

Crawley, A., Beynon, M., Munday, M., 2013. Making Location Quotients More Relevant as a Policy Aid in Regional Spatial Analysis. Urban Stud 50, 1854–1869. doi:10.1177/0042098012466601

Farole, T., Winkler, D., 2013. Firm location and the determinants of exporting in low- and middle-income countries. J Econ Geogr lbs060. doi:10.1093/jeg/lbs060

Fotheringham, A.S., 2000. Context-dependent spatial analysis: A role for GIS? Journal of Geographical Systems 2, 71–76. doi:10.1007/s101090050032

Geary, R.C., 1954. The Contiguity Ratio and Statistical Mapping. The Incorporated Statistician 5, 115. doi:10.2307/2986645

Gehlke, C.E., Biehl, K., 1934. Certain Effects of Grouping Upon the Size of the Correlation Coefficient in Census Tract Material. Journal of the American Statistical Association 29, 169. doi:10.2307/2277827

Grasland, C., Madelin, M., 2006. The Modifiable Area Unit Problem (Final Report No. 3.4.3). ESPON.

Grasland, C., Mathian, H. ’el\`ene, Vincent, J.-M., 2000. Multiscalar analysis and map generalisation of discrete social phenomena: Statistical problems and political consequences. Statistical Journal of the United Nations Economic Commission for Europe 17, 157–188.

Greenaway, D., Kneller, R., 2007. Firm heterogeneity, exporting and foreign direct investment*. The Economic Journal 117, F134–F161. doi:10.1111/j.1468-0297.2007.02018.x

Griffith, D.A., Chun, Y., 2018. GIS and Spatial Statistics/Econometrics: An Overview, in: Comprehensive Geographic Information Systems. Elsevier, pp. 1–26. doi:10.1016/B978-0-12-409548-9.09680-9

Huff, D.L., 1964. Defining and Estimating a Trading Area. Journal of Marketing 28, 34–38. doi:10.2307/1249154

Ince, D.C., Hatton, L., Graham-Cumming, J., 2012. The case for open computer programs. Nature 482, 485–488. doi:10.1038/nature10836

Isserman, A.M., 1980. Estimating Export Activity in a Regional Economy: A Theoretical and Empirical Analysis of Alternative Methods. International Regional Science Review 5, 155–184. doi:10.1177/016001768000500204

Koenig, P., 2009. Agglomeration and the export decisions of French firms. Journal of Urban Economics 66, 186–195. doi:10.1016/j.jue.2009.07.002

Lennert, M., 2015. The Use of Exhaustive Micro-Data Firm Databases for Economic Geography: The Issues of Geocoding and Usability in the Case of the Amadeus Database. ISPRS International Journal of Geo-Information 4, 62–86. doi:10.3390/ijgi4010062

Lennert, M., GRASS GIS Development Team, 2016. r.neighborhoodmatrix. Open Source Geospatial Foundation, USA.

Lennert, M., GRASS GIS Development Team, 2014. v.neighborhoodmatrix. Open Source Geospatial Foundation, USA.

Marcon, E., Puech, F., 2010. Measures of the geographic concentration of industries: improving distance-based methods. J Econ Geogr 10, 745–762. doi:10.1093/jeg/lbp056

Melitz, M.J., Redding, S.J., 2015. Chapter 1 - Heterogeneous Firms and Trade, in: Elhanan Helpman, K.R. and G.G. (Ed.), Handbook of International Economics, Handbook of International Economics. Elsevier, pp. 1–54.

Moran, P.A.P., 1950. Notes on Continuous Stochastic Phenomena. Biometrika 37, 17–23. doi:10.2307/2332142

Neteler, M., Bowman, M.H., Landa, M., Metz, M., 2012. GRASS GIS: A multi-purpose open source GIS. Environmental Modelling & Software 31, 124–130. doi:10.1016/j.envsoft.2011.11.014

Nosek, B.A., Alter, G., Banks, G.C., Borsboom, D., Bowman, S.D., Breckler, S.J., Buck, S., Chambers, C.D., Chin, G., Christensen, G., Contestabile, M., Dafoe, A., Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D.P., Hesse, B., Humphreys, M., Ishiyama, J., Karlan, D., Kraut, A., Lupia, A., Mabry, P., Madon, T., Malhotra, N., Mayo-Wilson, E., McNutt, M., Miguel, E., Paluck, E.L., Simonsohn, U., Soderberg, C., Spellman, B.A., Turitto, J., VandenBos, G., Vazire, S., Wagenmakers, E.J., Wilson, R., Yarkoni, T., 2015. Promoting an open research culture. Science 348, 1422–1425. doi:10.1126/science.aab2374

Openshaw, S., 1983. The modifiable areal unit problem. Geo Books, Norwick [Norfolk].

Redding, S.J., 2011. Theories of Heterogeneous Firms and Trade. Annual Review of Economics 3, 77–105. doi:10.1146/annurev-economics-111809-125118

Richardson, H.W., 1985. INPUT-OUTPUT AND ECONOMIC BASE MULTIPLIERS: LOOKING BACKWARD AND FORWARD*. Journal of Regional Science 25, 607–661. doi:10.1111/j.1467-9787.1985.tb00325.x

Strotebeck, F., 2010. The Location Quotient – Assembly and application of methodological enhancements [WWW Document]. URL http://mpra.ub.uni-muenchen.de/47988/ (accessed 10.21.14).

Taupier, R., Willis, C., 1994. Geographic Information Systems and Applied Economics: An Initial Discussion of Potential Applications and Contributions. Agricultural and Resource Economics Review 23, 140–149.

Tian, Z., 2013. Measuring Agglomeration Using the Standardized Location Quotient with a Bootstrap Method. Journal of Regional Analysis & Policy 43, 186–197.

Tiebout, C., 1956. Exports and Regional Economic Growth. Journal of Political Economy 64, 160–164. doi:10.1086/257771

Tobler, W.R., 1989. Frame independent spatial analysis, in: Goodchild, M.F., Gopal, S. (Eds.), The Accuracy of Spatial Databases. CRC Press, London, Great Britain, pp. 115–122.

Tobler, W.R., 1979. Smooth Pycnophylactic Interpolation for Geographical Regions. Journal of the American Statistical Association 74, 519–530. doi:10.2307/2286968

Tobler, W.R., 1970. A Computer Movie Simulating Urban Growth in the Detroit Region. Economic Geography 46, 234. doi:10.2307/143141

Zambelli, P., Gebbert, S., Ciolli, M., 2013. Pygrass: An Object Oriented Python Application Programming Interface (API) for Geographic Resources Analysis Support System (GRASS) Geographic Information System (GIS). ISPRS International Journal of Geo-Information 2, 201–219. doi:10.3390/ijgi2010201

Annexe

Calculation of the gamma coefficient in the Huff model

- For each NACE:
    - Calculate distance from each pixel in the total space to the closest production unit in that NACE
    - Calculate summary statistics (e.g. 90th percentile) of these distances
- Chose one of the summary statistics
- Determine the per-NACE γ as:
    - 0 if summary statistics of NACE is above a threshold max value of distance (i.e. distance does not really matter)
    - 2 if summary statistics of NACE is below a certain threshold min value (i.e. very local distribution of goods)
    - else: scale linearly between 0 and 2 according to the position of the summary statistic between the max and min threshold values