Southern Africa Labour and Development Research Unit

gpsbound: Routine for importing and verifying geographical information from a user provided shapefile

by
Tim S.L. Brophy, Reza Che Daniels
and
Sibongile Musundwa

Working Paper Series
Number 132

About the Author(s) and Acknowledgments

Tim Brophy is a Senior Data Analyst, NIDS, SALDRU, UCT, email: tslbrophy@gmail.com
Reza Che Daniels is a Senior Lecturer, School of Economics, UCT, email: reza.daniels@uct.ac.za
Sibongile Musundwa is a Research Assistant, NIDS, SALDRU, UCT, email: sibongile.musundwa@gmail.com

Recommended citation

Brophy, T.S., Daniels, R.C., Musundwa, S., (2014).gpsbound: Routine for importing and verifying geographical information from a user provided shapefile. A Southern Africa Labour and Development Research Unit Working Paper Number 132. Cape Town: SALDRU, University of Cape Town

---

ISBN: 978-1-920517-73-1

Working Papers can be downloaded in Adobe Acrobat format from www.saldru.uct.ac.za.
Printed copies of Working Papers are available for R15.00 each plus vat and postage charges.

Orders may be directed to:
The Administrative Officer, SALDRU, University of Cape Town, Private Bag, Rondebosch, 7701,
Tel: (021) 650 5696, Fax: (021) 650 5697, Email: brenda.adams@uct.ac.za

gpsbound: Routine for importing and verifying geographical information from a user provided shapefile*

Tim S.L. Brophy
University of Cape Town
Southern Africa Labour and Development Research Unit
Cape Town, South Africa
tslbrophy@gmail.com

Reza Che Daniels
University of Cape Town
School of Economics
Cape Town, South Africa
reza.daniels@uct.ac.za

Sibongile Musundwa
University of Cape Town
Southern Africa Labour and Development Research Unit
Cape Town, South Africa
sibongile.musundwa@gmail.com

07 July 2014

Abstract

Geographical coordinates such as Global Positioning System (GPS) latitude and longitude estimates form the foundation of many spatial statistical methods. gpsbound allows users to (1) import geographical information from the attribute table of a polygon shapefile based on the identified location of GPS coordinates in a Stata dataset, and (2) check that the GPS coordinates lie within the bounds of a polygon demarcated in the shapefile (e.g. enumerator areas, primary sampling units, suburb, city, country). One of the contributions of gpsbound is to allow users to work with spatial data in Stata without ever needing Geographical Information System (GIS) software.[1]

*This article has been accepted by the Stata Journal for publication.
1 The authors would like to thank participants at the 2013 Stata Conference New Orleans for valuable feedback on an earlier version of this algorithm called gpsmap. This name was subsequently changed to gpsbound due to the authors subsequently finding out that gpsmap is a registered trademark of the Garmin Corporation.

1 Introduction

Spatial data gives representation to features on earth. To accomplish this three data types are used: polygons, points and lines. Polygons describe a geographical area, such as the geopolitical boundaries of a country; points describe features represented by a single set of GPS coordinates, such as the location of a dwelling unit; and lines describe features such as the path of a road. Spatial data is most commonly stored in vector data format called a shapefile. A shapefile is a relational database that consists of multiple tables connected by a primary key (identifier). Each of the component tables have their own distinct data structure and file format. Traditionally, shapefiles are manipulated using Geographical Information System (GIS) software. `gpsbound` allows users to work with GPS coordinates and related information in Stata by manipulating selective components of the shapefile relational database.

Frequently, Stata datasets contain GPS coordinates for objects of interest (e.g. dwelling units, establishments), but do not have information about the geographical areas (polygons) into which those GPS coordinates fall. `gpsbound` is a routine in Stata that allows information from the attribute table of a user-provided polygon shapefile to be imported as new variables into the open (master) Stata dataset for each latitude and longitude pair. It also allows users to check that the GPS coordinates lie within a polygon of interest (e.g. enumerator areas, primary sampling units, suburb or county, city, country). If the GPS coordinates lie outside the polygon of interest, the algorithm treats those coordinates as errors of measurement and fails to import any data from the attribute table for affected points. This allows users to identify incorrect coordinates immediately which is useful for further diagnostic (and possibly data collection) work.

Before the introduction of `gpsbound` the process to import spatial data into Stata required a user to export their GPS coordinates of interest out of Stata into a text file, then import the text file into GIS software and convert the data into a point shapefile representing the GPS coordinates of interest. The user would then have to open a polygon shapefile in the GIS software and perform a spatial join between the point and the polygon shapefiles in order to link the points of interest to the attribute data of the polygon shapefile. These attribute table results would then need to be exported by the GIS software into a text file and imported by Stata for analysis.

With the introduction of `gpsbound` this process is no longer required. The user can simply specify the variables that contain decimal degree GPS coordinates and the file location of a polygon shapefile in decimal degree units, to which the coordinates need to be mapped. The algorithm will then perform the spatial join and import the relevant attribute data. Consequently, Stata users are no longer required to learn GIS software in order to be able to import geographical data for analysis.

While it was previously possible to import entire shapefiles and attribute tables into Stata using Kevin Crow's shp2data command, `gpsbound's` contribution is the selective importation of individually identified polygons and attribute data based on the GPS coordinates from a Stata dataset. We envisage this func-

tionality to have benefits to two types of Stata users: (1) researchers interested in importing the attribute data of polygons for given GPS coordinates; and (2) survey methodologists interested in verifying the location of sampled units (e.g. households, establishments) within selected enumerated areas or primary sampling units, which can be done (if necessary) in field in real time.

2 Practical experience with using gpsbound in a nationally representative longitudinal household survey

`gpsbound` was developed by the authors out of a practical problem that confronted us in the National Income Dynamics Study (NIDS), a nationally representative longitudinal household survey in South Africa. The first three waves of the survey were implemented by the Southern Africa Labour and Development Research Unit (SALDRU) at the University of Cape Town. The GPS coordinates of the dwelling units of respondents were recorded at each wave. The problem that confronted us was that (1) we needed a way to check that the selected dwelling units were indeed the correct ones in terms of the initial sample drawn in the first wave of the survey, and (2) in subsequent waves of the survey, we needed a way to validate that fieldworkers were either (a) revisiting the correct dwelling units, or (b) correctly recording the location of new dwelling units when continuing sample members of the survey had moved to new locations in the country and therefore new dwelling units.

In practical terms, fieldwork for Wave 3 of the survey ended in 2012. There were 9,273 dwelling units with observed GPS coordinates that were located across South Africa. `gpsbound` was used to (1) verify that the GPS coordinates were correctly recorded in field, and (2) after the completion of fieldwork, to import the geographical information for new dwelling units where previously no geographical information had been stored because they represented the dwelling units of individuals that had moved between waves of the survey.

Running `gpsbound` for 9,273 dwelling units on a standard PC with latitude and longitude coordinates resulted in the following computational times for different polygon shapefiles:

1. For the enumerator area (EA) shapefile from the South African National Census of 2011, which consisted of 103,576 EA polygons, the algorithm took 302 seconds;
2. For the sub-place shapefile (equivalent in geographical area terms to a suburb or small town), which consisted of 22,108 sub-place polygons, the algorithm took 186 seconds;
3. For the main-place shapefile (equivalent in geographical area terms to a big town or city), which consisted of 14,039 main-place polygons, the algorithm took 196 seconds;
4. For the district council shapefile (equivalent in geographical area terms to the size of a county), which consisted of 52 polygons, the algorithm took

76 seconds;

5. For the provinces shapefile, which consisted of 9 provinces, the algorithm took 310 seconds.

This non-linear trend in computational performance was due to the size of the shapefiles themselves, which differed according to the density of points per polygon.

3 Polygon spatial data and its file structure

A polygon shapefile consists of geographically demarcated borders with an attribute table. So, for example, a shapefile may consist of the provincial, city or town borders while the names of each regional unit would be stored in an attribute table. Therefore in order for a shapefile to be classified as such, there needs to be at least three mandatory file formats. (ESRI, 2014):

1. The shapefile (*.shp), which contains the spatial data describing the location of the features (e.g. GPS coordinates);
2. The dBase file (*.dbf), which contains a table of non-spatial attributes of the features (e.g. names of cities);
3. The index file (*.shx), which allows GIS software to effectively navigate the *.shp file.

It should be noted that the *.shp file extension is not the whole shapefile but a single component of the relational database that constitutes a shapefile. Consequently, it cannot be used independently of the other component files. In order to understand how `gpsbound` functions one needs to understand the different formats and types of data that are contained in the component files in more detail. As `gpsbound` uses polygon shapefiles we will limit our discussion to these.

3.1 The shapefile (*.shp)

Polygon shapefiles contain an index variable which is used as the primary key (identifier), as well as the x and y coordinates for a series of GPS coordinates. It also contains header information for each of the polygons. Polygon header data contains the location of each polygon within the shapefile as well as the number of GPS points that make up a polygon. Finally, it contains the bounding box data, which is the maximum and minimum x and y coordinates.

3.2 The dBase file (*.dbf)

The dBase file is a tabular file format containing the attribute data that describe the features of the shapefile as well as the index variable that will act as the primary key (identifier) to link the *.dbf and *.shp files in a one-to-many relationship.

3.3 The index file (*.shx)

The index file is, in non-technical terms, an indexing file that tells GIS software how to efficiently read the *.shp file. It does this by storing the position of the starting byte of each shapefile element, thus allowing the GIS software to quickly navigate to specific elements within a shapefile without having to process the whole shapefile.

4 The gpsbound command

4.1 Description

`gpsbound` maps decimal degree coordinate points to a decimal degree polygon shapefile. For each set of latitude and longitude coordinates, it returns the attributes of the polygon within which that point lies. An optional variable indicating whether the GPS coordinates were mapped successfully is also returned.

It is important to note that `gpsbound` works only with decimal degree latitude and longitude coordinates and polygon shapefiles. If the GPS coordinates are in any other format (e.g. degrees-minutes-seconds), they have to be converted to decimal degrees first before `gpsbound` can be used.

4.2 Syntax

The syntax for `gpsbound` is:

`gpsbound` *using shapefilename* [*if*] , *latitude(varname) longitude(varname)* [
`valid`(*newvarname*) `prefix`(*string*) `keepusing`(*varlist*)]

`gpsbound` requires two inputs: the polygon shapefile and the variable names which contain the latitude and longitude coordinates. The full path and name of the shapefile must be included, including the file suffix (*.shp). An associated dBase (*.dbf) file must also be stored in the same folder path as the shapefile and have the same name as the shapefile, though the file suffix will differ.

4.3 Options

The following options are available for use:

`valid`(*newvar*) is an optional binary variable created to indicate whether the latitude and longitude coordinates that identify a point (e.g. dwelling unit) fall within the bounds of the correct polygon (e.g. enumerated area). A new variable name is required by this option.

`prefix`(*string*) option requires a string input. This string input will be used as a prefix and will be incorporated to the imported variable names that are obtained from the attribute table of the shapefile.

`keepusing`(*varlist*) option requires a list of variable names to be identified from the attribute table of the shapefile. These variables can be viewed by opening the *.dbf file in a spreadsheet programme. Only the variables listed in keepusing will then be imported from the shapefile. If this option is not selected all variables from the shapefile attribute table will be imported.

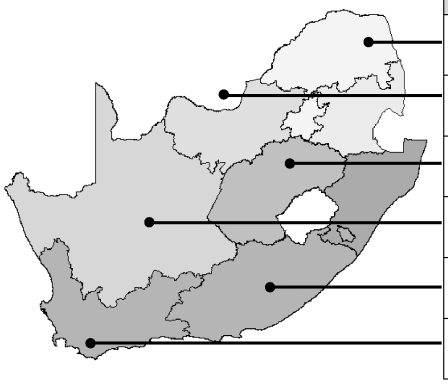
4.4 Output

`gpsbound` returns output in two forms: the first is the imported variable(s) from the shapefile attribute table, which are added to the open dataset; and the second is an optional binary variable that indicates the success of the mapping routine between the GPS point and the polygon shapefile. This latter output is only created when the option valid is selected.

▷ Example

The illustration below is a map (and associated polygon shapefile) of the 9 provinces of the Republic of South Africa, the Kingdom of Lesotho and the Kingdom of Swaziland (the Kingdoms are the non-shaded polygons within the contiguous geopolitical boundary of South Africa). The map also identifies six sets of decimal degree GPS coordinates for randomly selected points in and around South Africa. Five of the randomly selected points fall within the bounds of South Africa and the remaining point falls just outside of the borders of South Africa, inside the neighbouring country Botswana.

There are two columns in the table indicating the results we expect to see once we run `gpsbound`. The third column in the table represents the expected outcome of the optional output valid which, as discussed above, is a binary variable indicating whether the given GPS coordinates of interest fall within any of the provincial polygons that make up South Africa's nine provinces. The fourth column then indicates the expected value of the Province' attribute that will be returned for each point by running `gpsbound` given the set of coordinates and the South African province shapefile.



Latitude	Longitude	Expected "Valid" Value	Expected Province
-23.302522	30.499787	1	Limpopo
-25.243937	24.975272	0	
-27.333873	27.507088	1	Free State
-29.354913	21.882635	1	Northern Cape
-32.194854	26.393136	1	Eastern Cape
-34.290047	19.816570	1	Western Cape

Figure 1: Example of gpsbound

```
. * CREATE THE EXAMPLE DATASET
. clear
. set obs 6
obs was 0, now 6
. gen Latitude = .
(6 missing values generated)
. replace Latitude =      -23.302522      in      1
(1 real change made)
. replace Latitude =      -25.243937      in      2
(1 real change made)
. replace Latitude =      -27.333873      in      3
(1 real change made)
. replace Latitude =      -29.354913      in      4
(1 real change made)
. replace Latitude =      -32.194854      in      5
(1 real change made)
. replace Latitude =      -34.290047      in      6
(1 real change made)
. gen Longitude = .
(6 missing values generated)
. replace Longitude =     30.499787       in      1
(1 real change made)
. replace Longitude =     24.975272       in      2
(1 real change made)
. replace Longitude =     27.507088       in      3
(1 real change made)
. replace Longitude =     21.882635       in      4
(1 real change made)
. replace Longitude =     26.393136       in      5
(1 real change made)
. replace Longitude =     19.81657        in      6
(1 real change made)
. * RUN GPSBOUND
. * Run gpsbound on the South African province shapefile available
. * at http://www.demarcation.org.za/[2]
```

[2]No direct URL is available to the download page. On the home page click the download menu, select Boundary Data the boundary data download page will then appear. Clicking the Province folder will download the Shapefile Zip.

```
.
. gpsbound using "C:\Users\User\Desktop\Province\Province_New_SANeighbours.shp",
.       latitude(Latitude longitude(Longitude) valid(valid) keepusing(PROVINCE)
```

Execution of the above yields the following table of results:

```
. list
```

	Latitude	Longitude	valid	PROVINCE
1.	-23.30252	30.49979	1	Limpopo
2.	-25.24394	24.97527	0	
3.	-27.33387	27.50709	1	Free State
4.	-29.35491	21.88264	1	Northern Cape
5.	-32.19485	26.39314	1	Eastern Cape
6.	-34.29005	19.81657	1	Western Cape

As can be seen the actual results and the expected results are the same, demonstrating that `gpsbound` functions as desired.

◁

5 Methods and Formulas

In discussing `gpsbound`, it is important to have an understanding of the actions that are taken by `gpsbound` to produce the final result of mapping, validating and importing the attribute table of the user provided shapefile into Stata.

`gpsbound` was written to be used on decimal degree latitude and longitude coordinates with the World Geodetic Systems 1984 datum (WGS 84). A geodetic datum translates GPS coordinates into their relative position on earth. The need for geodetic datum arises because of the earth's ellipsoid shape. WGS 84 is the geodetic datum used by modern day GPS units and satellite navigation systems. It requires the latitude coordinate to fall between -90 and 90 degrees inclusive and requires the longitude coordinates to fall between -180 and 180 degrees inclusive.(Spatial Reference (2007)) To this end the algorithm checks to ensure that the coordinates being passed to `gpsbound` fall within this range. A similar check is performed on the polygon shapefile.

5.1 gpsbound subroutines

Broadly speaking, the `gpsbound` routine can be broken down into six distinct actions powered by six different subroutines. The first subroutine imports the dBase (.dbf) data file. The dBase file contains the geographical attribute table for each of the polygons that make up the user provided shapefile. The second subroutine imports the shapefile headers. What the headers are and what they contain will be explored later in the paper. This is followed by the third subroutine, which identifies potential polygons into which each set of GPS coordinates under investigation may fall. The fourth subroutine imports each of the polygons that were identified as a potential polygon in subroutine three

(Crow (2006)). This is then followed by a fifth subroutine which runs a point-in-polygon algorithm to associate each point with its corresponding polygon.

The fifth subroutine which is the point-in-polygon algorithm returns two key sets of results to Stata. The first is a binary variable indicating if the given GPS point falls within any of the potential polygons. If so, a second data record is returned, which contains the unique identifier for each polygon. The sixth and last subroutine merges the attribute table imported by the first subroutine into the dataset. This is done based on the polygon identifier returned from the point-in-polygon algorithm.

5.2 Importing the dBase (.dbf) attribute table

As outlined, the first subroutine performed by `gpsbound` involves importing the dBase file. The dBase file in its own rights is a fully fledged file format (dBASE (2014)). The simplicity of dBase's data structure means that it has been adopted by many other applications to assist in managing and storing data. In the context of shapefiles, the dBase files are used to store the feature attributes for each of the polygons that are contained within a shapefile. There is only one requirement for the use of a dBase file as data format for shapefile attributes, and that is that a unique identifier field is included. The values in the identifier field in the dBase file must correspond to the values in the identifier field in the shapefile. Other than that, all the fields contained in the attribute table are defined by the author of the shapefile, which allows the author to attribute any characteristics they deem necessary to a particular shapefile. These attributes can be global attributes applicable to each shape in a shapefile or they can be unique for each shape in a shapefile. This is the real power of shapefiles and their attributes tables. Thus one of the goals of `gpsbound` is to access this attribute information and link it to given GPS coordinates collected in field or through other operations. The linking of the shapefile attributes allows researchers to access geographical information for analysis within Stata. As there is no limit to the information attributable to polygons contained within a shapefile the information that can be derived and analysed from the use of shapefiles and GPS coordinates through `gpsbound` is only limited by the availability of GIS data.

5.3 Importing the shapefile headers

The second subroutine imports the headers of the polygons from the shapefile. A shapefile can be broken up into two distinct categories of data, namely the headers and the coordinate data that describes each polygon's shape. The header data occurs for each polygon contained in the shapefile. Thus if a user provided shapefile contains one hundred individual polygons then the shapefile will contain 100 headers and 100 x n rows of coordinates, where n is the number of coordinate points that make up a polygon's shape. The headers contain basic information pertaining to each of the polygons such as its primary key or identification number, the GPS coordinates of its bounding box as well as the position of its starting and ending bits in the shapefile (Environmental Systems

Research Institute (ESRI) (1998)). Thus the second major action of `gpsbound` is to read through the shapefile and retrieve each of the polygon headers. The headers are then stored in a matrix that we call the headers matrix. This matrix is held in Mata and is crucial to the efficiency of both the computational time taken when running `gpsbound` and the amount of memory needed to run `gpsbound`. Memory efficiency is of particular concern when using shapefiles because they can contain hundreds of millions of coordinate points which if all held in memory at the same time affects both the overall computing speed and available memory for other applications as well as Stata itself. [3]

5.4 Identifying possible polygons

It is at this point that we start to process each set of GPS coordinates passed by the user to `gpsbound` via the command's longitude and latitude parameters. Each GPS coordinate is compared to the bounding boxes found in the headers matrix. The bounding box of a polygon is defined by the minimum and maximum latitude and longitude of that polygon; in other words, it is a box that fits exactly around the polygon. If the user-provided latitude and longitude falls within the bounding box of a particular polygon, then that polygon is considered a possible polygon into which the coordinates might fall. This exercise will result in a list of possible polygon matches for each of the coordinates provided. This list of possible polygons is used to limit the number of iterations, so that each point only has to be tested against a few polygons and not against all the polygons in the polygon shapefile.

5.5 Importing possible polygons

Possible polygons that were identified for each GPS point by subroutine three are imported into Mata through the use of a shapefile polygon reader. The reader imports a specifically identified polygon.

5.6 The point-in-polygon algorithm

Each polygon is then subjected to the point-in-polygon algorithm until the GPS point is definitively placed inside one of the polygons. If however the GPS point cannot be definitively placed inside any of the polygons a binary variable containing a zero value is generated indicating that the point was not matched to the shapefile at all.

The point-in-polygon algorithm makes use of a ray casting approach and the even-odd rule. The even-odd rule states that a ray drawn from the point of interest to infinity will intersect the edge of the polygon an odd number of times if that point falls within the polygon (The World Wide Web Consortium (W3C) (2011)). However if the point of interest falls outside of the polygon

[3] Previous versions of `gpsbound` did not have this step but rather read all of the shapefile data into Mata; this was found to be computationally inefficient and used an unnecessary amount of memory.

then the ray will intersect the border of the polygon an even number of times. This is best explained using illustrations.

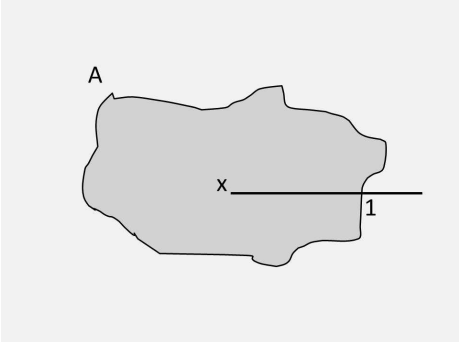


Figure 2: Illustration1

In illustration 1 the polygon is represented by A with X being the point of interest. Casting a horizontal ray to infinity, the ray intersects the edge of polygon A at intersection 1.

As can be seen, the ray only intersects the edge of the polygon once at intersection 1. Applying the even-odd rule here tells us that the point of interest X falls within the polygon A as the numbers of intersections with the edge of the polygon A was odd.

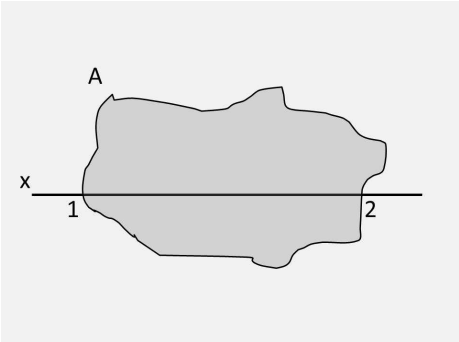


Figure 3: Illustration2

In illustration 2 the point of interest X falls outside of the polygon A. As can be seen from the illustration; a ray cast from the point of interest X intersects the polygon A twice, namely at 1 and 2.

Applying the even-odd rule, point X would be determined to fall outside of polygon A as the horizontal ray intersects the edges of polygon A an even number of times.

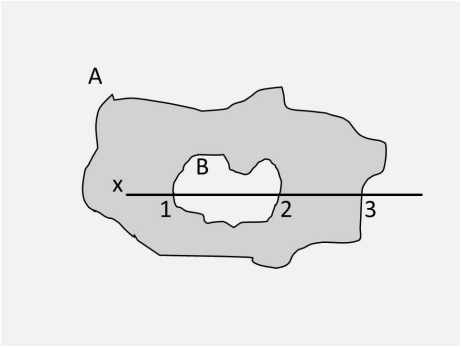


Figure 4: Illustration3

Illustration 3 demonstrates polygon A with a separate polygon B inside it. Given point of interest X, if we draw a horizontal ray away from that point, the ray will intersect the edge of polygon three times at 1, 2 and 3. Applying the even-odd rule we can tell that point X lies inside of polygon A as there is an odd number of intersections with the edges of polygon B.

In order to find the number of intersections of the horizontal ray with the polygon edge, we break the polygon edge into segments, where each segment is a straight line connecting one point on the edge to the next point. We need to consider only those segments of the polygon edge which cross the latitude of the point of interest. That is to say, we consider those segments where one endpoint has a larger Y coordinate than the point of interest, and the other endpoint has a smaller Y coordinate.

We can then divide these segments into three classes: class A consists of those segments where both endpoints lie to the right of (i.e. have a larger X coordinate than) the point of interest; class B consists of those where both endpoints lie to the left of (i.e. have a smaller X coordinate than) the point of interest; and class C consists of those where one endpoint lies to the left and the other endpoint lies to the right of the point of interest.

Recall that we want to count the number of intersections of the polygon edge with the horizontal ray that starts at the point of interest and projects rightwards to infinity. Edge segments in class A definitely intersect this ray, while edge segments in class B definitely do not intersect this ray. Edge segments in class C may or may not intersect this ray, and require further testing. As we are using Mata for the point-in-polygon algorithm it is more efficient to calculate the intersects of the segments in both class A and C as a matrix operation as opposed to subsampling them into two distinct groups first. In so doing we calculate all the points where the line segments of both A and C cross the latitude of the point of interest. If this crossing point is to the right of the point of interest, then the segment does intersect the ray. To do this we determine the straight line equation that describes the segment, and then substitute our Y coordinate from our point of interest into the equation to determine the X coordinate of the crossing point.

Once we have calculated all the points of intersection for both class A and B we then count how many elements of our results matrix are greater than or

equal to the X coordinate of our point of interest.

As a final step our count is determined to be either even or odd and this result is returned by the algorithm. Where the algorithm returns the result as odd, the index of that shape file is returned to Stata. This is then used to merge in the attribute table that was imported earlier.

5.7 Joining the attribute data to the GPS coordinates

Finally the two data results, namely the binary variable indicating if mapping to the shapefile was possible and a second variable containing the shapefile polygon identification number, are returned to Stata. It is at this point that the dBase file that was imported into Stata by subroutine one is merged to the mapping results on the shapefile polygon identification number. It needs to be noted that in versions of Stata before Stata 13 the string variables from the dBase file are limited to 244 characters in length as this is the Stata limit on the length of string variables. From Stata 13 onwards this is no longer a limitation due to the introduction of the strL data type by Stata. BOB

5.8 Saved Results

There are no saved results in `gpsbound` because if the GPS coordinate is validated then a new variable(s) is imported into the Stata dataset in memory directly from the attribute table of the shapefile.

6 Conclusion

Spatial data enables a wide range of statistical analyses to be conducted. The ability to selectively import additional spatial information into datasets from shapefiles, combined with the ability to check the accuracy of GPS coordinates, is crucial for more advanced statistical methods to be utilised. `gpsbound` makes an important contribution to the Stata user community in this respect. Not only does it enable researchers to increase the scope of plausible analytical methods to be applied to spatial datasets, but it also enables survey methodologists to verify in-field operational concerns in real time, such as checking that the correct EAs have been visited by fieldworkers and/or that all dwelling units sampled lie within the correct EAs. It is important to note that for `gpsbound` to operate correctly, GPS coordinates must be recorded as, or converted to, decimal degree format, and that shapefiles must be polygon shape files rather than point or line shapefiles. With these basic inputs, the full functionality of `gpsbound` can be realised.

7 Acknowledgements

We are grateful to staff members at the National Income Dynamics Study at the Southern Africa Labour and Development Research Unit for providing the time and space to adequately test the algorithm. We are especially grateful to Adrian Frith for his invaluable comments on earlier drafts of this paper. Lastly, we would like to thank participants at the annual Stata conference in New Orleans, 2013 for helpful comments on `gpsbound`.

8 gpsbound Ado and help file

Should you wish to obtain a copy of the Ado and the help file, please use "findit gpsbound" in Stata or alternately contact the authors.

References

Crow, K. (2006), 'SHP2DTA: Stata module to converts shape boundary files to Stata datasets', Statistical Software Components, Boston College Department of Economics.
URL: *http://ideas.repec.org/c/boc/bocode/s456718.html*

dBASE (2014), 'Data file header structure for the dbase version 7 table file'. [Accessed: 2014, April 07].
URL: *http://www.dbase.com/Knowledgebase/INT/db7_file_fmt.htm*

Environmental Systems Research Institute (ESRI) (1998), Esri shapefile technical description, Technical report, Environmental Systems Research Institute, Redlands.

Spatial Reference (2007), 'Epsg:4326 : Wgs 84'. [Accessed: 2014, April 07].
URL: *http://spatialreference.org/ref/epsg/wgs-84/*

The World Wide Web Consortium (W3C) (2011), 'Painting: Filling, stroking and marker symbols'. [Accessed: 2014, April 07].
URL: *http://www.w3.org/TR/SVG/painting.html#FillProperties*

southern africa labour and development research unit

The Southern Africa Labour and Development Research Unit (SALDRU) conducts research directed at improving the well-being of South Africa's poor. It was established in 1975. Over the next two decades the unit's research played a central role in documenting the human costs of apartheid. Key projects from this period included the Farm Labour Conference (1976), the Economics of Health Care Conference (1978), and the Second Carnegie Enquiry into Poverty and Development in South Africa (1983-86). At the urging of the African National Congress, from 1992-1994 SALDRU and the World Bank coordinated the Project for Statistics on Living Standards and Development (PSLSD). This project provide baseline data for the implementation of post-apartheid socio-economic policies through South Africa's first non-racial national sample survey.

In the post-apartheid period, SALDRU has continued to gather data and conduct research directed at informing and assessing anti-poverty policy. In line with its historical contribution, SALDRU's researchers continue to conduct research detailing changing patterns of well-being in South Africa and assessing the impact of government policy on the poor. Current research work falls into the following research themes: post-apartheid poverty; employment and migration dynamics; family support structures in an era of rapid social change; public works and public infrastructure programmes, financial strategies of the poor; common property resources and the poor. Key survey projects include the Langeberg Integrated Family Survey (1999), the Khayelitsha/Mitchell's Plain Survey (2000), the ongoing Cape Area Panel Study (2001-) and the Financial Diaries Project.

www.saldru.uct.ac.za
Level 3, School of Economics Building, Middle Campus, University of Cape Town
Private Bag, Rondebosch 7701, Cape Town, South Africa
Tel: +27 (0)21 650 5696
Fax: +27 (0) 21 650 5797
Web: www.saldru.uct.ac.za