

# geoplot: A new command to draw maps

Ben Jann

University of Bern

2023 UK Stata Conference  
London, September 7–8, 2023

# Outline

- 1 Introduction
- 2 Syntax
- 3 Examples
- 4 Conclusions

# Introduction

- Official Stata has limited support for drawing maps.<sup>1</sup>
- The command most people use is `spmap` by Maurizio Pisati.<sup>2</sup>
- `spmap` is wonderful, but it also has its limitations.
- This is why I wrote a new command to draw maps; the new command is called `geoplot`.

---

<sup>1</sup>Although Stata's `graph twoway` does provide the basic building blocks needed for drawing maps.

<sup>2</sup>Pisati's `spmap` has been integrated into official Stata as command `grmap` at some point; it can be activated by typing `grmap, activate`. Functionality appears to be identical to `spmap`.

# Frames

- A main challenge with maps is that, typically, the data is scattered across multiple files.
  - ▶ For example, different types of features (e.g. borders, lakes, points of interest, etc.) are usually kept in separate files.
  - ▶ Furthermore, in many cases, two files are used to store the data of a given set of units.
    - ★ An attribute file: one row per unit containing an ID and several attribute variables.
    - ★ A shape file: multiple rows per unit containing polygon coordinates.
- `geoplot` addresses this challenge by using **frames** (requires Stata 16 or newer). The main idea is to treat data management and plotting as two separate tasks.
  1. Command `geoframe` loads the data into frames (and possibly performs various other data management tasks).
  2. Command `geoplot` then draws the map. Linkages between frames will be handled automatically in the background.

# Some guiding principles

- Managing the data should be convenient and intuitive. The data management toolbox should be easy to expand.
- The graph command should follow Stata's `graph` syntax as much as possible.
- Different layers of objects should be combinable in any order.
- The available set of layer types should be easy to expand.
- In general: make life as easy as possible for users.

1 Introduction

2 Syntax

3 Examples

4 Conclusions

# geoframe - prepare the data

```
[frame frame:] geoframe subcommand [...]
```

<i>subcommand</i>	Description
<b>Main</b>	
<a href="#">create</a>	load data into geoframe or declare current frame as geoframe
<a href="#">link</a>	link shape frame to current frame
<a href="#">clean</a>	delete unmatched/empty shapes and units
<a href="#">select</a>	select units and shapes
<a href="#">describe</a>	describe geoframe
<b>Manipulation</b>	
<a href="#">generate</a>	generate special-purpose variable in current frame
<a href="#">spjoin</a>	spatially join points in current frame to shapes from other frame
<a href="#">bbox</a>	store bounding box, enclosing circle, or convex hull in new frame
<b>Settings</b>	
<a href="#">set</a>	update geoframe settings of current frame
<a href="#">get</a>	retrieve geoframe settings from current frame
<b>Utilities</b>	
<a href="#">rename</a>	rename a geoframe
<a href="#">duplicate</a>	duplicate a geoframe
<a href="#">relink</a>	fix linkage variable after modifying data
<a href="#">unlink</a>	unlink shape frame from current frame
<a href="#">attach</a>	attach attribute frame to current frame using aliases (Stata 18 required)
<a href="#">detach</a>	detach attribute frame from current frame (Stata 18 required)
<a href="#">copy</a>	copy variables from attribute frame to current frame
<a href="#">append</a>	append observations from other frame to current frame

# geoplot - draw a map

```
geoplot (layer) [(layer) ...] [, global_options]
```

where *layer* is

```
layertype [frame] [...] [, options]
```

<i>layertype</i>	Description
<b>area</b>	shapes, potentially filled
<b>line</b>	shapes, line only
<b>point</b>	single-coordinate markers
<b>label</b>	single-coordinate labels
<b>symbol</b>	single-coordinate symbols (circles, hexagons, stars, etc.)
* <b>pie</b>	pie charts
* <b>bar</b>	stacked bar charts
<b>pcspike</b>	paired-coordinate spikes
<b>pccapsym</b>	paired-coordinate spikes capped with symbols
<b>pcarrow</b>	paired-coordinate arrows
<b>pcbarrow</b>	paired-coordinate arrows with two heads
<b>pcpoint</b>	paired-coordinate markers
* <b>pointi</b>	<b>point</b> with immediate arguments
* <b>pci</b>	<b>pcspike</b> with immediate arguments
* <b>pcarrowi</b>	<b>pcarrow</b> with immediate arguments
* <b>symboli</b>	<b>symbol</b> with immediate arguments



# geoplot - draw a map

A key feature is that in most layer types an auxiliary variable can be specified (argument *zvar*) to affect the rendering of the plotted elements (colors, line widths, marker symbols, etc.).

<i>zvar_options</i>	Description
<hr/>	
Main	
<a href="#"><code>discrete</code></a>	treat <i>zvar</i> as discrete instead of continuous
<a href="#"><code>levels(spec)</code></a>	number of levels and method to determine cuts
<a href="#"><code>cuts(numlist)</code></a>	use levels defined by specified cuts
<a href="#"><code>colorvar([i.]zvar)</code></a>	alternative to specifying <i>zvar</i> as argument
Styling	
* <a href="#"><code>color(palette)</code></a>	colors
* <a href="#"><code>lwidth(list)</code></a>	line widths
* <a href="#"><code>lpattern(list)</code></a>	line patterns
* <a href="#"><code>fintensity(list)</code></a>	fill intensities
* <a href="#"><code>msymbol(list)</code></a>	marker symbols
* <a href="#"><code>msize(list)</code></a>	marker sizes
* <a href="#"><code>msangle(list)</code></a>	marker angles
* <a href="#"><code>mlwidth(list)</code></a>	marker outline widths
* <a href="#"><code>mlabsize(list)</code></a>	marker label sizes
* <a href="#"><code>mlabangle(list)</code></a>	marker label angles
* <a href="#"><code>mlabcolor(palette)</code></a>	marker label colors
Legend keys	
* <a href="#"><code>label(spec)</code></a>	set labels of legend keys and related settings
Missing	
<a href="#"><code>missing(options)</code></a>	styling of elements for which <i>zvar</i> is missing

---

1 Introduction

2 Syntax

3 Examples

4 Conclusions

## Step 1: Download data

- GIS boundary files covering Greater London (file “statistical-gis-boundaries-london.zip” from [data.london.gov.uk](http://data.london.gov.uk)).
- Strategic Industrial Location Points (file “lp-consultation-oct-2009-sil-points-shp.zip” [data.london.gov.uk](http://data.london.gov.uk)).
- London Ward Well-Being Scores (file “london-ward-well-being-probability-scores.xls” from [data.london.gov.uk](http://data.london.gov.uk)).
- Road Safety Data (file “dft-road-casualty-statistics-accident-2021” from [www.data.gov.uk](http://www.data.gov.uk)).
- Shape file of River Thames from [github.com/geothery/londonShapefiles](https://github.com/geothery/londonShapefiles).

Copyright statements:

Contains National Statistics data © Crown copyright and database right 2012

Contains Ordnance Survey data © Crown copyright and database right 2012

## Step 2: Prepare data for use in Stata

- Use official Stata's `spshape2dta` to transform shape files to Stata format.
- For example, “statistical-gis-boundaries-london.zip” contains the following files.

```
. ls, wide
LSOA_2004_London_Low_Resolution.dbf
LSOA_2004_London_Low_Resolution.prj
LSOA_2004_London_Low_Resolution.shp
LSOA_2004_London_Low_Resolution.shx
LSOA_2011_London_gen_MHW.dbf
LSOA_2011_London_gen_MHW.prj
LSOA_2011_London_gen_MHW.sbn
LSOA_2011_London_gen_MHW.sbx
LSOA_2011_London_gen_MHW.shp
LSOA_2011_London_gen_MHW.shp.xml
LSOA_2011_London_gen_MHW.shx
London_Borough_Excluding_MHW.GSS_CODE.atx
London_Borough_Excluding_MHW.NAME.atx
London_Borough_Excluding_MHW.dbf
London_Borough_Excluding_MHW.prj
London_Borough_Excluding_MHW.sbn
London_Borough_Excluding_MHW.sbx
etc...
```

## Step 2: Prepare data for use in Stata

- Now apply `spsshape2dta`; I am only interested in Boroughs and Wards.

```
. spshape2dta London_Borough_Excluding_MHW, saving(Borough)
  (importing .shp file)
  (importing .dbf file)
  (creating _ID spatial-unit id)
  (creating _CX coordinate)
  (creating _CY coordinate)
file Borough_shp.dta created
file Borough.dta      created

. spshape2dta London_Ward_CityMerged.cpg, saving(Ward)
  (importing .shp file)
  (importing .dbf file)
  (creating _ID spatial-unit id)
  (creating _CX coordinate)
  (creating _CY coordinate)
file Ward_shp.dta created
file Ward.dta      created
```

- The shape files of River Thames and SIL points can be processed in a similar way.

## Step 2: Prepare data for use in Stata

- Now also import the accident data (csv) and the well-being scores (xls).

```
// accidents
import delimited dft-road-casualty-statistics-accident-2021.csv, clear
destring location_osgr, gen(_X) force
destring location_northing_osgr, gen(_Y) force
keep accident_index _X _Y
save Accidents

// well-being
import excel london-ward-well-being-probability-scores.xls, sheet(Data) ///
    clear allstring firstrow
drop if Newwardcode==" "
qui destring LifeExpectancy20052009-BL, replace force
rename Newwardcode GSS_CODE
local nm Absenteeism
rename UnauthorisedAbsenceinAllScho `nm'2009
rename (AD AP AQ AR) (`nm'2010 `nm'2011 `nm'2012 `nm'2013)
local nm Children_ofwhh
rename dependentchildreninoutofw `nm'2009
rename (AT AU AV AW) (`nm'2010 `nm'2011 `nm'2012 `nm'2013)
local nm PublicTrans
rename PublicTransportAccessibility `nm'2009
rename (AY AZ BA BB) (`nm'2010 `nm'2011 `nm'2012 `nm'2013)
local nm Openspace
rename Homeswithhaccesstoopenspace `nm'2009
rename (BD BE BF BG) (`nm'2010 `nm'2011 `nm'2012 `nm'2013)
local nm Wellbeing
rename Subjectivewellbeingaveragesc `nm'2009
rename (BI BJ BK BL) (`nm'2010 `nm'2011 `nm'2012 `nm'2013)
save Wellbeing
```

## Step 3: Load data into frames using geoframe

- Load the data on wards using geoframe create.

```
. geoframe create Ward
(reading shapes from Ward_shp.dta)
(all observations in frame Ward_shp matched)
(link to frame Ward_shp added)
(current frame now Ward)

      Frame name: Ward
      Frame type: unit
      Feature type: <none>
      Number of obs: 625
      Unit ID: _ID
      Coordinates: _CX _CY
      Area: <none>
      Linked shape frame: Ward_shp
```

## Step 3: Load data into frames using geoframe

- When loading an attribute file, geoframe looks for an associated shape file (*filename\_shp.dta* in same folder) and loads it into a second frame and links the two frames. Here is the description of the additional frame:

```
. geoframe describe Ward_shp
      Frame name: Ward_shp
      Frame type: shape
      Feature type: <none>
      Number of obs: 158,520
      Unit ID: _ID
      Coordinates: _X _Y
      Within-unit sort ID: shape_order
      Within-unit polygon ID: <none>
      Plot level ID: <none>
```

- Add attributes from the well-being dataset using merge.

```
. merge 1:1 GSS_CODE using Wellbeing, keep(match master) nogenerate
```

Result	Number of obs
Not matched	0
Matched	625



## Step 3: Load data into frames using geoframe

- Use same procedure to load the data on boroughs.

```
. geoframe create Borough
(reading shapes from Borough_shp.dta)
(all observations in frame Borough_shp matched)
(link to frame Borough_shp added)
(current frame now Borough)

      Frame name: Borough
      Frame type: unit
      Feature type: <none>
      Number of obs: 33
      Unit ID: _ID
      Coordinates: _CX _CY
      Area: <none>
      Linked shape frame: Borough_shp

. merge 1:1 GSS_CODE using Wellbeing, keep(match master) nogenerate
```

Result	Number of obs
Not matched	0
Matched	33

## Step 3: Load data into frames using geoframe

- For the SIL data, the shape file is redundant (each shape is just a single point). Specify `noshp` to omit the shape file

```
. geoframe create SIL, noshp
(current frame now SIL)

      Frame name: SIL
      Frame type: unit
      Feature type: <none>
      Number of obs: 59
      Unit ID: _ID
      Coordinates: _CX _CY
      Area: <none>
      Linked shape frame: <none>
```

- Loading the shape file would, in fact, not hurt (apart from wasting a bit of working memory). So `noshp` is not strictly necessary.

## Step 3: Load data into frames using geoframe

- For the Thames data, the attribute file is redundant (just a single unit; no extra variables), so I directly load the shape data (again, loading both files would not hurt).

```
. geoframe create Thames using Thames_shp, feature(water)
(current frame now Thames)

      Frame name: Thames
      Frame type: shape
      Feature type: water
      Number of obs: 3,017
      Unit ID: _ID
      Coordinates: _X _Y
      Within-unit sort ID: shape_order
      Within-unit polygon ID: <none>
      Plot level ID: <none>
```

- Option `feature(water)` declares the type of feature included in the frame; this will be picked up by `geoplot`.

## Step 3: Load data into frames using geoframe

- For the accidents data there is only an attribute file (no shape file).

```
. geoframe create Accidents
(current frame now Accidents)
      Frame name: Accidents
      Frame type: unit
      Feature type: <none>
      Number of obs: 101,087
      Unit ID: <none>
      Coordinates: _X _Y
      Area: <none>
      Linked shape frame: <none>
```

## Step 4: Draw a map using geoplot

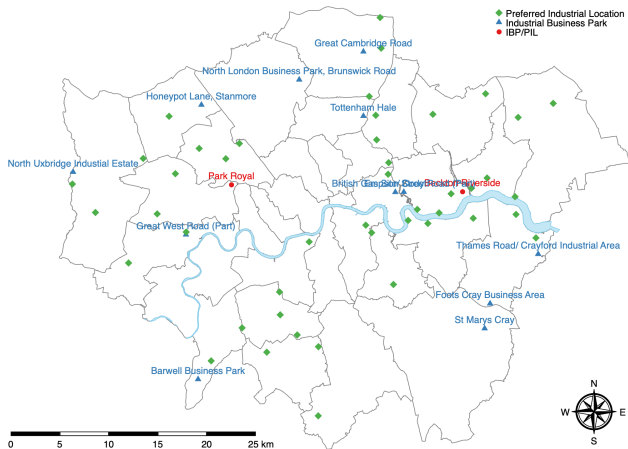
- Boroughs, wards, and river Thames.

```
. geoplot (area Ward) (line Borough, lwidth(.35)) (area Thames), tight
```



# Add some points of interest and other stuff

```
. frame change SIL
. encode SES_Type, generate(Type)
. geoplot (line Borough) (area Thames) (point SIL i.Type, ms(o t d)) ///
>   (label SIL Location i.Type if Type<=2, size(vsmall) pos(12)) ///
>   , tight margin(l=12) sbar(units(km)) compass
```



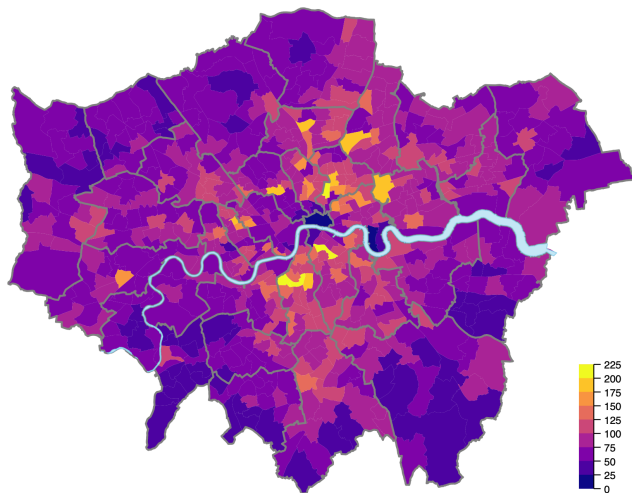
# Custom symbols

```
. geplot (line Borough) (area Thames) ///  
> (symbol SIL if Type==3, shape(pin) angle(-25) color(Teal) size(*.5)) ///  
> (symbol SIL if Type==2, shape(pentagram) color(sand) size(*.5)) ///  
> (symbol SIL if Type==1, shape(pin2) color(red)), tight
```



# Add color depending on attribute

```
. geoplot (area Ward Crimerate2013, color(plasma) cuts(0(25)225)) ///  
>      (line Borough, lwidth(.4)) (area Thames), tight clegend(position(se))
```

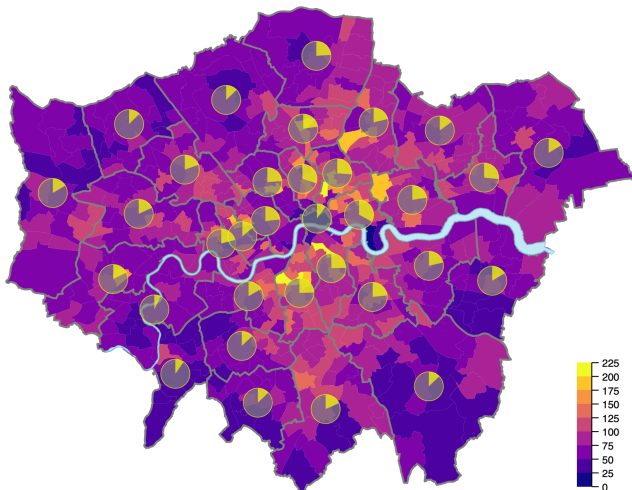


(crime rate 2013)



## Add second attribute using pie chart

```
. geoplot (area Ward Crimerate2013, color(plasma) cuts(0(25)225)) ///  
> (line Borough, lwidth(.4)) (area Thames) ///  
> (pie Borough Children_ofwhh2013, color(Yellow%70) asis ///  
> outline(fc(gray%70) below)), tight clegend(position(se))
```

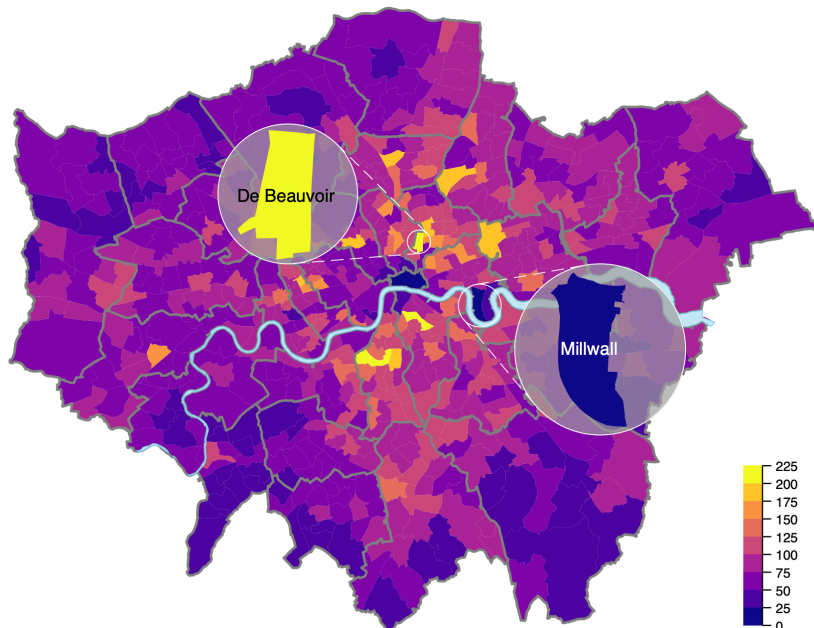


(percentage of dependent children in out-of-work households 2013)

# Zoom in on min and max

```
. frame change Ward
. su Crimerate2013, meanonly
. su _ID if inlist(Crimerate2013,r(min),r(max)), meanonly
. local min = r(min)
. local max = r(max)
. geoplot ///
>   (area Ward Crimerate2013, cuts(0(25)225) col(plasma)) ///
>   (line Borough, lwidth(.4)) ///
>   (area Thames) ///
>   (area Ward Crimerate2013, cuts(0(25)225) col(plasma) select(_ID==`min') ///
>     box(circle pad(5) fc(gs10%70))) ///
>   (label Ward NAME if _ID==`min', color(white)) ///
>   (area Ward Crimerate2013, cuts(0(25)225) col(plasma) select(_ID==`max') ///
>     box(circle pad(5) fc(gs10%70))) ///
>   (label Ward NAME if _ID==`max', color(black)) ///
>   , tight clegend(pos(se)) ///
>   zoom(4/5:4 150 -20, circle connect(lp(dash)) lcolor(white)) ///
>   zoom(6/7:6 200 160, circle connect(lp(dash)) lcolor(white))
```

# Zoom in on min and max



# Accidents

```
. geoplot (point Accidents, msymbol(p))
```



# Accidents

```
. frame Accidents: geoframe spjoin Borough  
(plevel not set; assuming that there are no nested polygons)  
(77974 points not matched)  
(variable _ID added to frame Accidents)  
. geoplot (line Borough) (area Thames) ///  
> (point Accidents if _ID<., msymbol(p) pstyle(p2)), tight
```

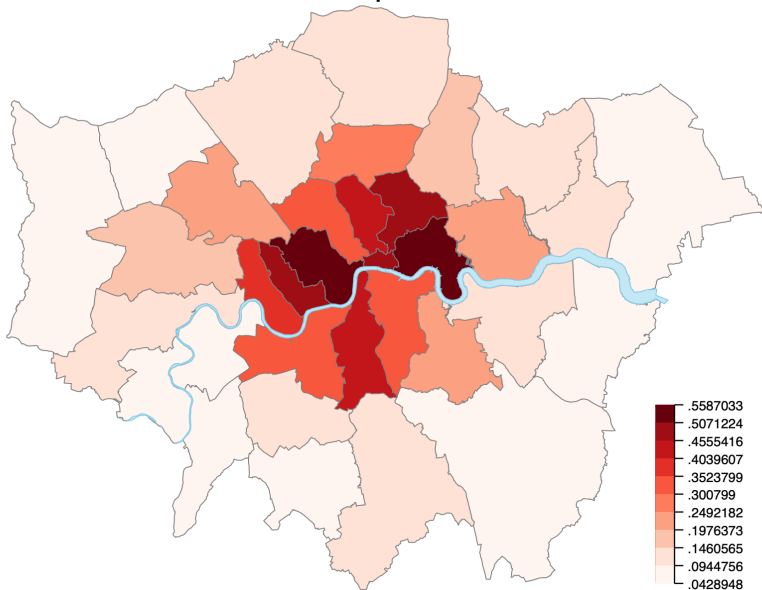


# Accidents

```
. frame copy Accidents temp
. frame change temp
. keep if _ID<.
(77,991 observations deleted)
. contract _ID
. frame change Borough
. geoframe copy temp _freq
(all observations in frame Borough matched)
(1 variable copied from linked frame)
. frame drop temp
. generate AccidentDensity = _freq / (HECTARES - NONLD_AREA)
. geoplot (area Borough AccidentDensity, levels(10) color(ReDs)) ///
>   (line Borough) (area Thames), tight clegend(position(se)) ///
>   title(Accidents per Hectare)
```

(Note that command `geoframe contract` could be used to obtain the accident count by borough with less typing.)

# Accidents per Hectare



1 Introduction

2 Syntax

3 Examples

4 Conclusions



# Conclusions

- `geoplot` provides a powerful and (relatively) easy to use toolbox for creating maps in Stata.
- Install from [github.com/benjann/geoplot](https://github.com/benjann/geoplot) or from SSC (`palettes`, `colrspace`, and `moremata` required).
- Thanks to Asjad Naqvi for extensive testing and many valuable suggestions; also see Asjad's post on `geoplot` at [medium.com/the-stata-guide](https://medium.com/the-stata-guide).
- Some future plans
  - ▶ Add support for shapefile translation and projections.
  - ▶ More spatial algorithms (buffering, clipping, joining, generalizing, collapsing, ...)
  - ▶ Add support for bivariate maps (see `bimap` by Asjad).
  - ▶ ...