geoplot: A new command to draw maps in Stata

Ben Jann

University of Bern

Workshop on "Creating maps with Stata and thematic applications"
Hermosillo, October 24–25, 2023

Outline

Introduction

- Official Stata has limited support for drawing maps.[1]
- The command most people use is `spmap` by Maurizio Pisati.[2]
- `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`.

[1] Although Stata's `graph twoway` does provide the basic building blocks needed for drawing maps.

[2] 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.

geoframe - prepare the data

[frame *frame*:] geoframe *subcommand* [...]

subcommand	Description
Main	
translate	translate shapefile source to Stata format (without loading)
convert	synonym for **geoframe translate**
create	load data into geoframe or declare current frame as geoframe
link	link shape frame to current frame
clean	delete unmatched/empty shapes and units
query	obtain information on shapes in geoframe
describe	describe geoframe
Manipulation	
project	apply projection
select	select units and shapes
clip	clip shapes using convex window
rclip	clip shapes using rectangular window
simplfy	simplify (generalize) shapes
bshare	select shared borders
generate	generate special-purpose variables
copy	copy variables between frames
append	append observations between frames

geoframe - prepare the data

```
Spatial join
  collapse        collapse points from other frame into current frame
  contract        contract points from other frame into current frame
  spjoin          match points in current frame to shapes from other frame

Generate shapes
  grid            store grid lines in new frame
  bbox            store bounding box, enclosing circle, or convex hull in new frame
  symbol          generate symbol shapes and store in new frame
  symboli         geoframe symbol with immediate arguments

Settings
  set             update geoframe settings of current frame
  get             retrieve geoframe settings from current frame

Utilities
  rename          rename a geoframe
  duplicate       duplicate a geoframe
  relink          fix linkage variable after modifying data
  unlink          unlink shape frame from current frame
  attach          attach attribute frame to current frame using aliases (Stata 18 required)
  detach          detach attribute frame from current frame (Stata 18 required)
```

geoplot - draw a map

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

where *layer* is:

layertype [*frame*] [...] [, *options*]

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

zvar_options	Description
Main	
discrete	treat *zvar* as discrete instead of continuous
levels(*spec***)**	number of levels and method to determine cuts
cuts(*numlist***)**	use levels defined by specified cuts
colorvar([i.]*zvar***)**	alternative to specifying *zvar* as argument
Styling	
* **color(***palette***)**	colors
* **lwidth(***list***)**	line widths
* **lpattern(***list***)**	line patterns
* **fintensity(***list***)**	fill intensities
* **msymbol(***list***)**	marker symbols
* **msize(***list***)**	marker sizes
* **msangle(***list***)**	marker angles
* **mlwidth(***list***)**	marker outline widths
* **mlabsize(***list***)**	marker label sizes
* **mlabangle(***list***)**	marker label angles
* **mlabcolor(***palette***)**	marker label colors
Legend keys	
* **label(***spec***)**	set labels of legend keys and related settings
nolegend	do not consider the layer for the default legend
Missing	
missing(*options***)**	styling of elements for which *zvar* is missing

Step 1: Download data

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

Step 2: Convert data for use in Stata

- Use `geoframe convert` (or official Stata's `spshape2dta`) to transform shape files to Stata format.
- For example, "statistical-gis-boundaries-london.zip" contains the following files.

```
. ls data/statistical-gis-boundaries-london/ESRI/, 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: Convert data for use in Stata

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

```
. local path data/statistical-gis-boundaries-london/ESRI/
. geoframe convert Borough using `path'London_Borough_Excluding_MHW, replace
  (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
  (type geoframe create Borough to load the data)
. geoframe convert Ward using `path'London_Ward_CityMerged, replace
  (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
  (type geoframe create Ward to load the data)
```

Step 2: Convert data for use in Stata

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

```
. geoframe convert SIL data/lp-consultation-oct-2009-sil-points-shp/, replace
(translating lp-consultation-oct-2009-sil-points from directory data/lp-consultation-
> oct-2009-sil-points-shp/)
  (importing .shp file)
  (importing .dbf file)
  (creating _ID spatial-unit id)
  (creating _CX coordinate)
  (creating _CY coordinate)
  file SIL_shp.dta created
  file SIL.dta     created
  (type geoframe create SIL to load the data)
. geoframe convert Thames data/londonShapefiles-master/inst/external/river_thames.dbf
> , replace
  (importing .shp file)
  (importing .dbf file)
  (creating _ID spatial-unit id)
  (creating _CX coordinate)
  (creating _CY coordinate)
  file Thames_shp.dta created
  file Thames.dta     created
  (type geoframe create Thames to load the data)
```

Step 2: Convert data for use in Stata

- Now also convert data on accidents (csv) and well-being scores (xls).

```
. // accidents
. import delimited data/dft-road-casualty-statistics-accident-2021.csv, clear
(encoding automatically selected: UTF-8)
(36 vars, 101,087 obs)
. destring location_easting_osgr, gen(_X) force
location_easting_osgr: contains nonnumeric characters; _X generated as long
(17 missing values generated)
. destring location_northing_osgr, gen(_Y) force
location_northing_osgr: contains nonnumeric characters; _Y generated as long
(17 missing values generated)
. keep accident_index _X _Y
. save Accidents, replace
file Accidents.dta saved
. // well-being
. import excel data/london-ward-well-being-probability-scores.xls, ///
>       sheet(Data) clear allstring firstrow
(64 vars, 711 obs)
. drop if Newwardcode==""
(52 observations deleted)
. qui destring *, replace
. rename Newwardcode GSS_CODE
. save Wellbeing, replace
file Wellbeing.dta saved
```

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), 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>
```

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
. geoframe describe Borough_shp
            Frame name: Borough_shp
            Frame type: shape
          Feature type: <none>
         Number of obs: 48,584
               Unit ID: _ID
           Coordinates: _X _Y
   Within-unit sort ID: shape_order
Within-unit polygon ID: <none>
         Plot level ID: <none>
```

Step 3: Load data into frames using geoframe

- The attribute files on wards and boroughs do not really contain much information that would be substantively interesting. For example, here is the contents of the wards frame:

```
. frame Ward: describe
Contains data from Ward.dta
 Observations:           625
    Variables:            10                  23 Oct 2023 03:21
```

Variable name	Storage type	Display format	Value label	Variable label
_ID	int	%12.0g		Spatial-unit ID
_CX	double	%10.0g		x-coordinate of area centroid
_CY	double	%10.0g		y-coordinate of area centroid
NAME	str37	%37s		NAME
GSS_CODE	str9	%9s		GSS_CODE
HECTARES	double	%12.3f		HECTARES
NONLD_AREA	double	%12.3f		NONLD_AREA
LB_GSS_CD	str9	%9s		LB_GSS_CD
BOROUGH	str22	%22s		BOROUGH
POLY_ID	long	%11.0f		POLY_ID

```
Sorted by: _ID
     Note: Dataset has changed since last saved.
```

Step 3: Load data into frames using geoframe

- So a next step typically is to add some substantive data from an alternative source. In our case, this is the data on well-being scores. The data can be merged into the attribute frames by variable `GSS_CODE`, which contains the ID code of the ward or borough.
- We could use official Stata's command `merge` for that purpose. However, we can also load the data into memory using `geoframe create` and then merge data using `geoframe copy`.
- I prefer the second approach because it allows me to load the data into working memory and manipulate them on the fly before merging.
- The variables from the well-being data I am interested in are called `Crimerate2013` ("Crime rate in 2013") and `AW` ("% dependent children in out-of-work households in 2013").

Step 3: Load data into frames using geoframe

```
. geoframe create Wellbeing, nodescribe
(current frame now Wellbeing)
. rename Crimerate2013 Crimerate
. rename AW Jobless
. frame Ward: geoframe copy Wellbeing Crimerate Jobless, id(GSS_CODE)
(all units matched)
(2 variables copied from frame Wellbeing)
. frame Borough: geoframe copy Wellbeing Crimerate Jobless, id(GSS_CODE)
(all units matched)
(2 variables copied from frame Wellbeing)
. frame Ward: describe
Contains data from Ward.dta
 Observations:           625
    Variables:            12                  23 Oct 2023 03:21
```

Variable name	Storage type	Display format	Value label	Variable label
_ID	int	%12.0g		Spatial-unit ID
_CX	double	%10.0g		x-coordinate of area centroid
_CY	double	%10.0g		y-coordinate of area centroid
NAME	str37	%37s		NAME
GSS_CODE	str9	%9s		GSS_CODE
HECTARES	double	%12.3f		HECTARES
NONLD_AREA	double	%12.3f		NONLD_AREA
LB_GSS_CD	str9	%9s		LB_GSS_CD
BOROUGH	str22	%22s		BOROUGH
POLY_ID	long	%11.0f		POLY_ID
Crimerate	double	%10.0g		Crime rate - 2013
Jobless	double	%10.0g		% dependent children in out-of-work households - 2013

```
Sorted by: _ID
```

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>
```

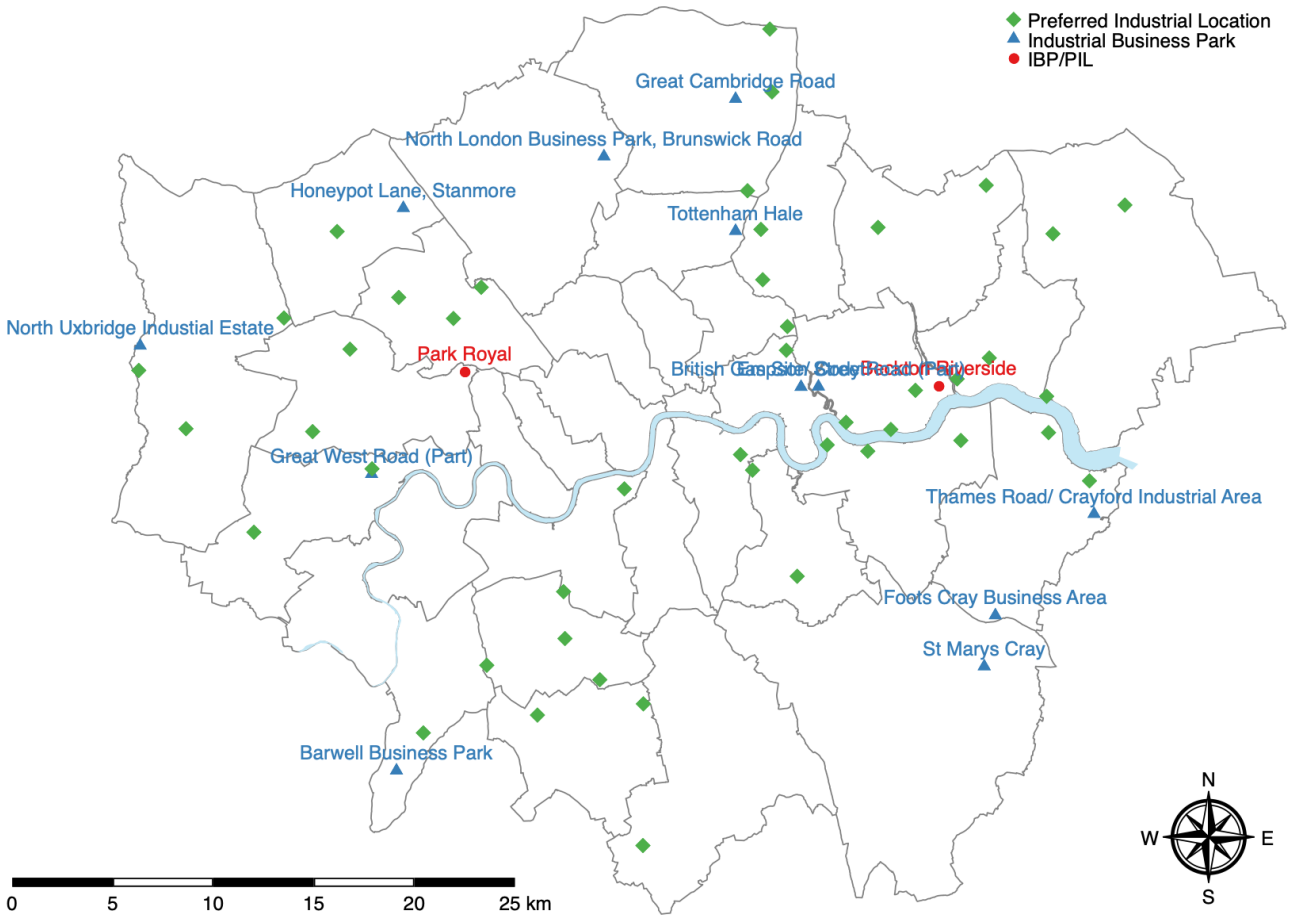
Step 4: Draw a map using geoplot

- 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
```

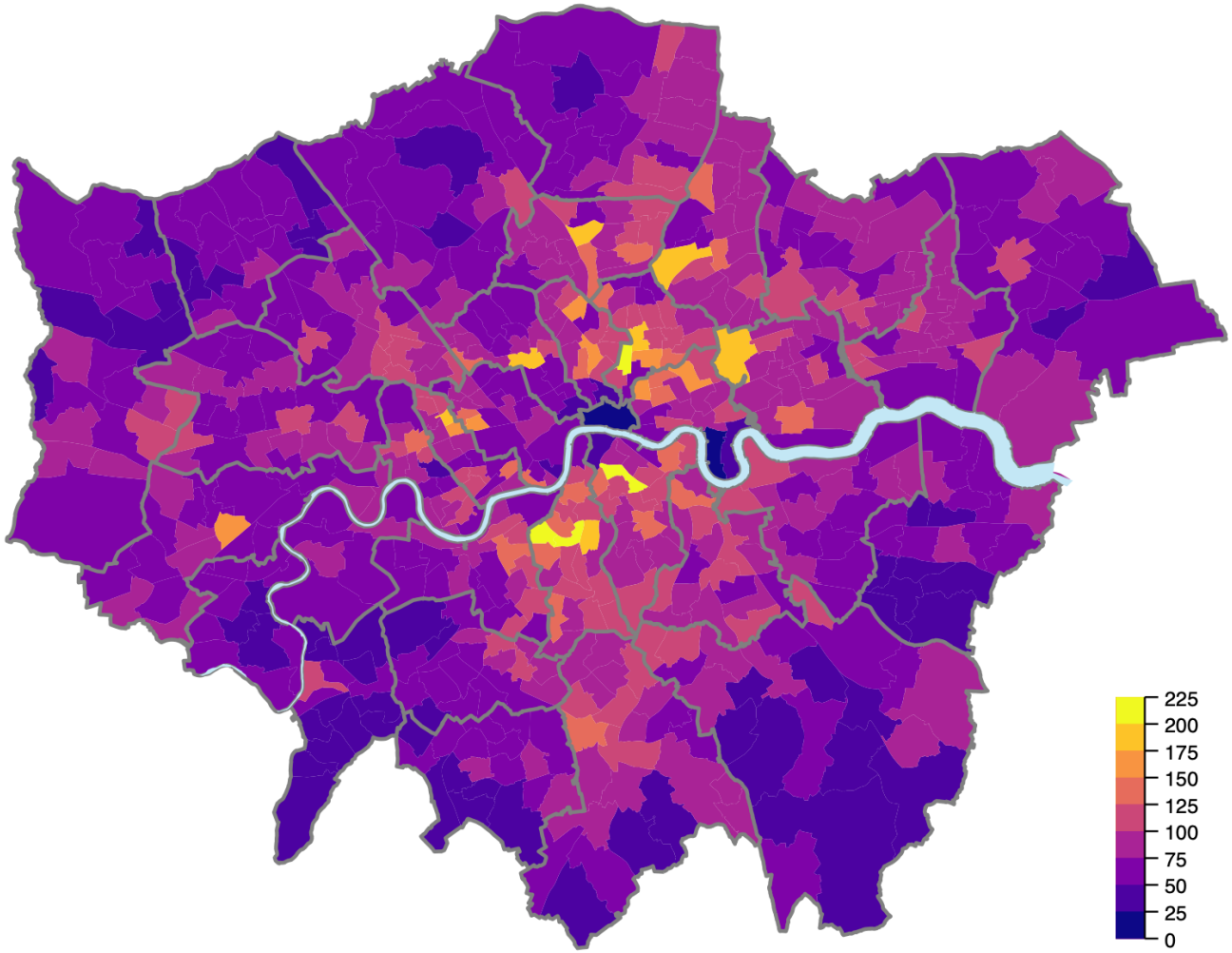


Custom symbols

```
. geoplot (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

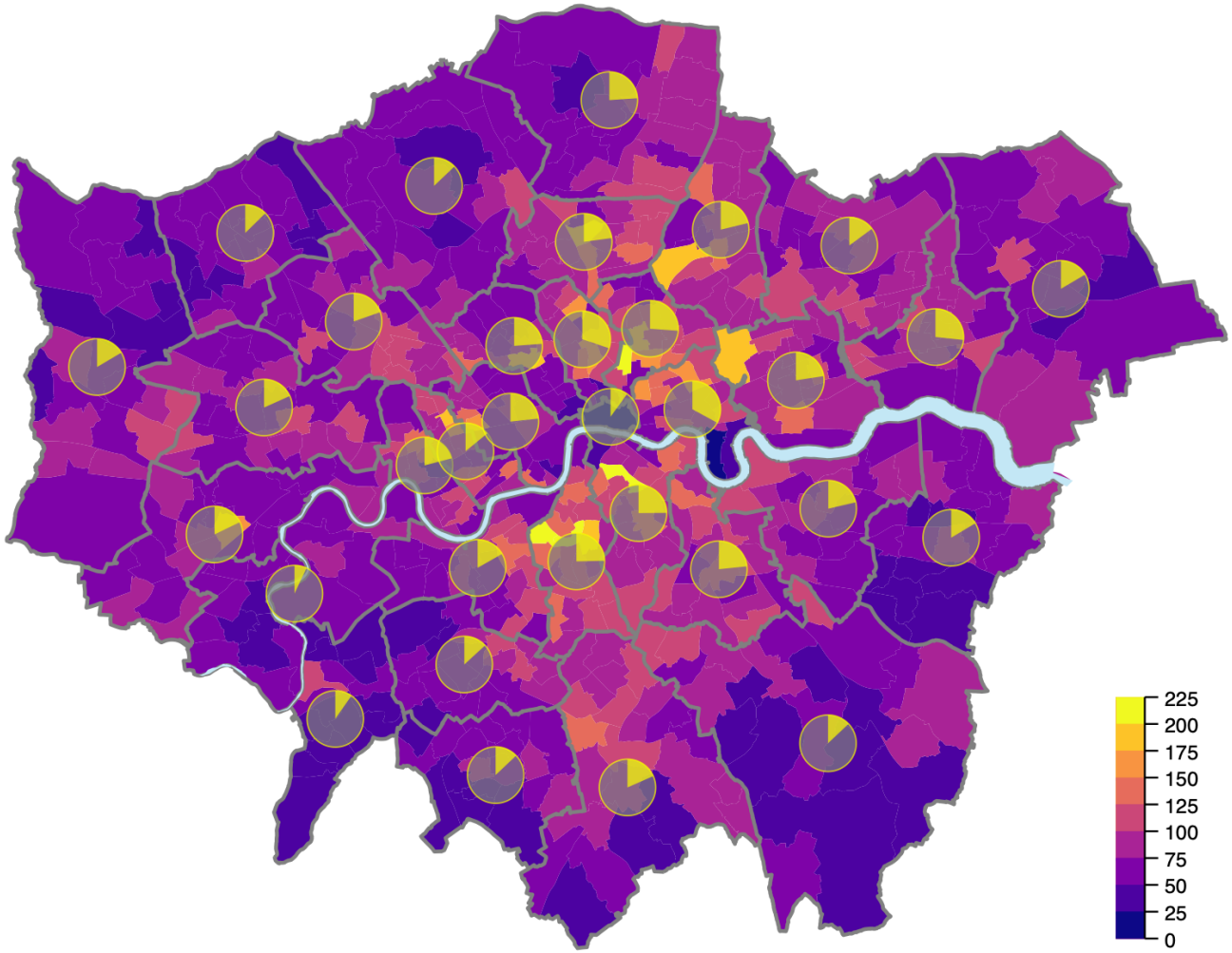
```
. geoplot (area Ward Crimerate, color(plasma) cuts(0(25)225)) ///
>     (line Borough, lwidth(.4)) (area Thames), tight clegend(position(se))
```



(crime rate 2013)

Add second attribute using pie chart

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

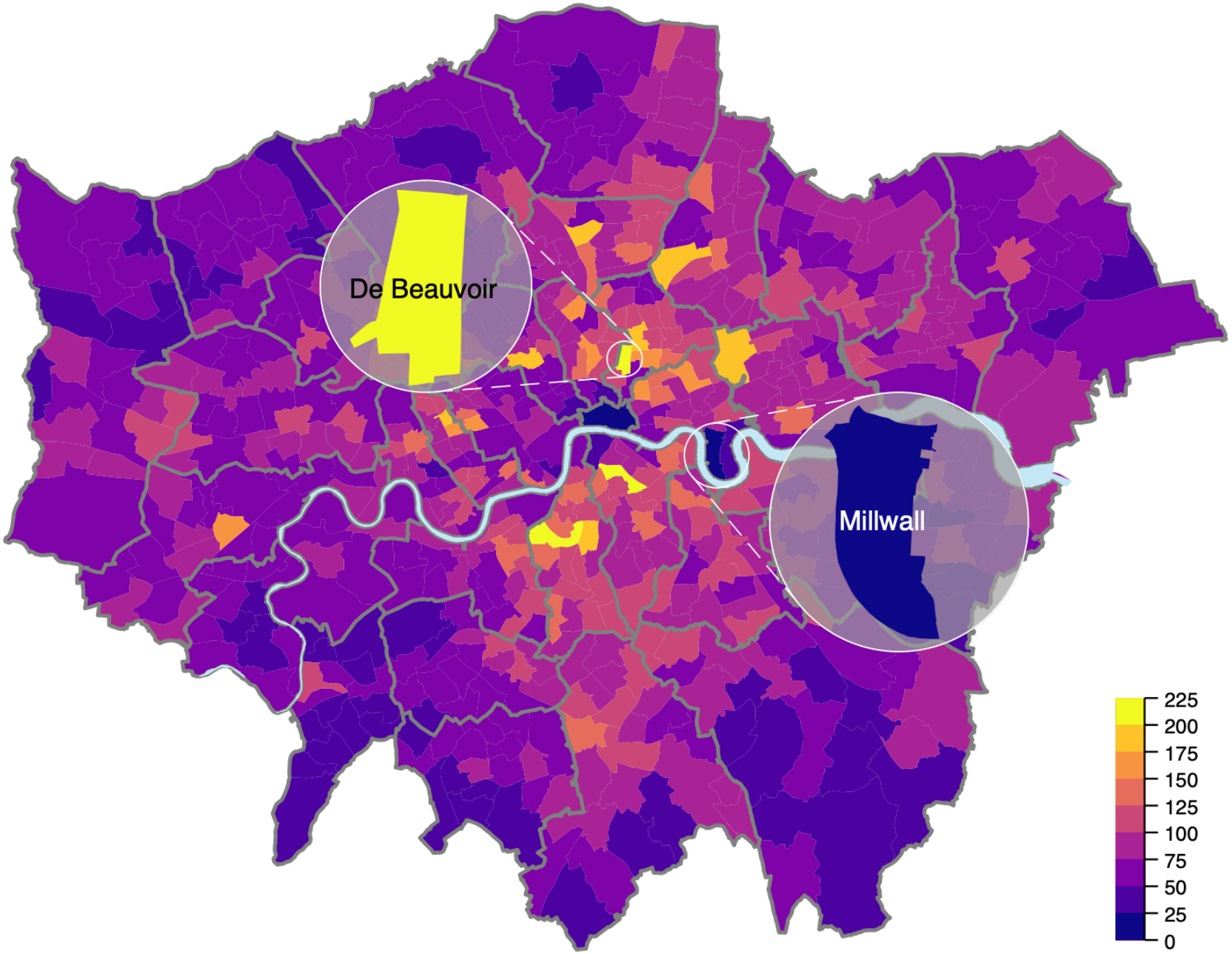


(percentage of dependent children in jobless households 2013)

Zoom in on min and max

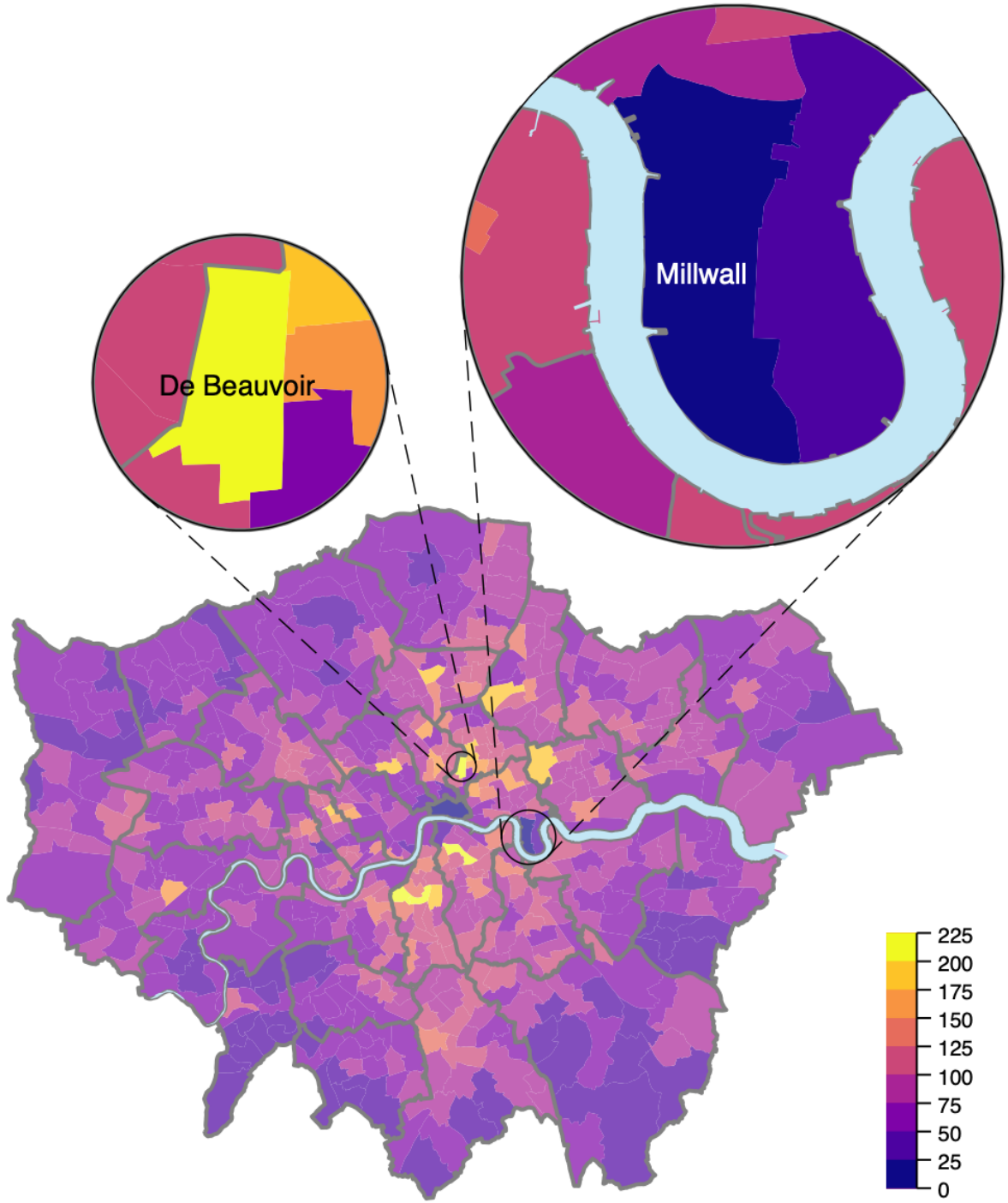
```
. frame change Ward
. su Crimerate, meanonly
. su _ID if inlist(Crimerate,r(min),r(max)), meanonly
. local min = r(min)
. local max = r(max)
. geoplot ///
>     (area Ward Crimerate, cuts(0(25)225) col(plasma)) ///
>     (line Borough, lwidth(.4)) ///
>     (area Thames) ///
>     (area Ward Crimerate, 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 Crimerate, 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



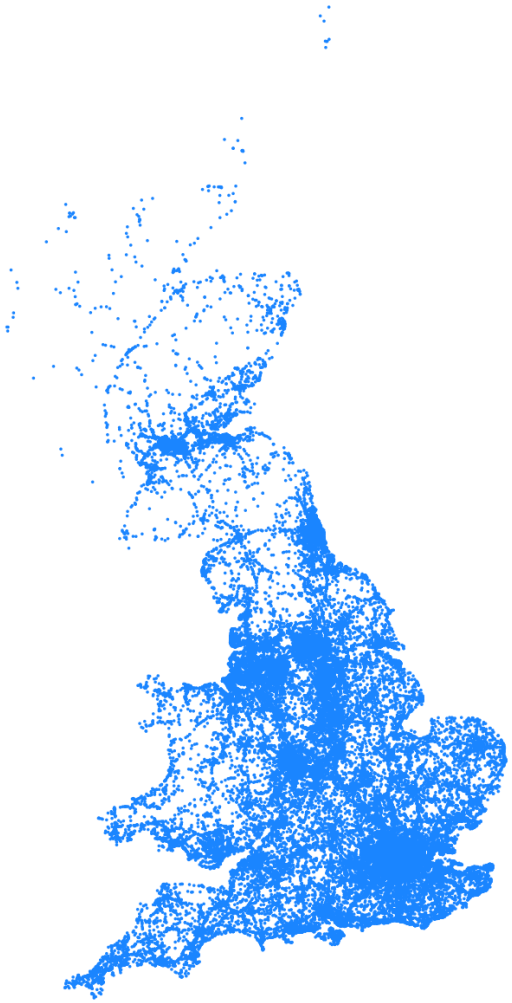
Zoom with clipped shapes

```
frame change Ward
su Crimerate, meanonly
su _ID if inlist(Crimerate,r(min),r(max)), meanonly
local min = r(min)
local max = r(max)
geoframe query bbox if _ID==`min', pad(20) n(50) circle
mat min = r(bbox)
geoframe query bbox if _ID==`max', pad(20) n(50) circle
mat max = r(bbox)
foreach frame in Ward Borough Thames {
    frame `frame': geoframe clip min, into(`frame'_min)
    frame `frame': geoframe clip max, into(`frame'_max)
}
geoplot ///
    (area Ward Crimerate, color(plasma, intensity(0.7)) cuts(0(25)225)) ///
    (line Borough, lwidth(.4)) ///
    (area Thames) ///
    (area Ward_min Crimerate, color(plasma) cuts(0(25)225)) ///
    (line Borough_min, lwidth(.4)) ///
    (area Thames_min) ///
    (label Ward NAME if _ID==`min', color(white)) ///
    (area Ward_max Crimerate, color(plasma) cuts(0(25)225)) ///
    (line Borough_max, lwidth(.4)) ///
    (area Thames_max) ///
    (label Ward NAME if _ID==`max', color(black)) ///
    , tight clegend(layer(4) position(se)) ///
      zoom(4/7:10 220 70, circle connect(lp(dash)) lcolor(black)) ///
      zoom(8/11:10 300 120, circle connect(lp(dash)) lcolor(black))
```


Millwall
De Beauvoir
225
200
175
150
125
100
75
50
25
0

Accidents

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



Accidents

- Data seems to be on entire Great Britain. So let's select the appropriate portion of the data.
- I use `geoframe` `spjoin` to spatially join the accident data with boroughs. This will add the borough ID to the accident data, which can then be used to select the appropriate points when plotting the data.

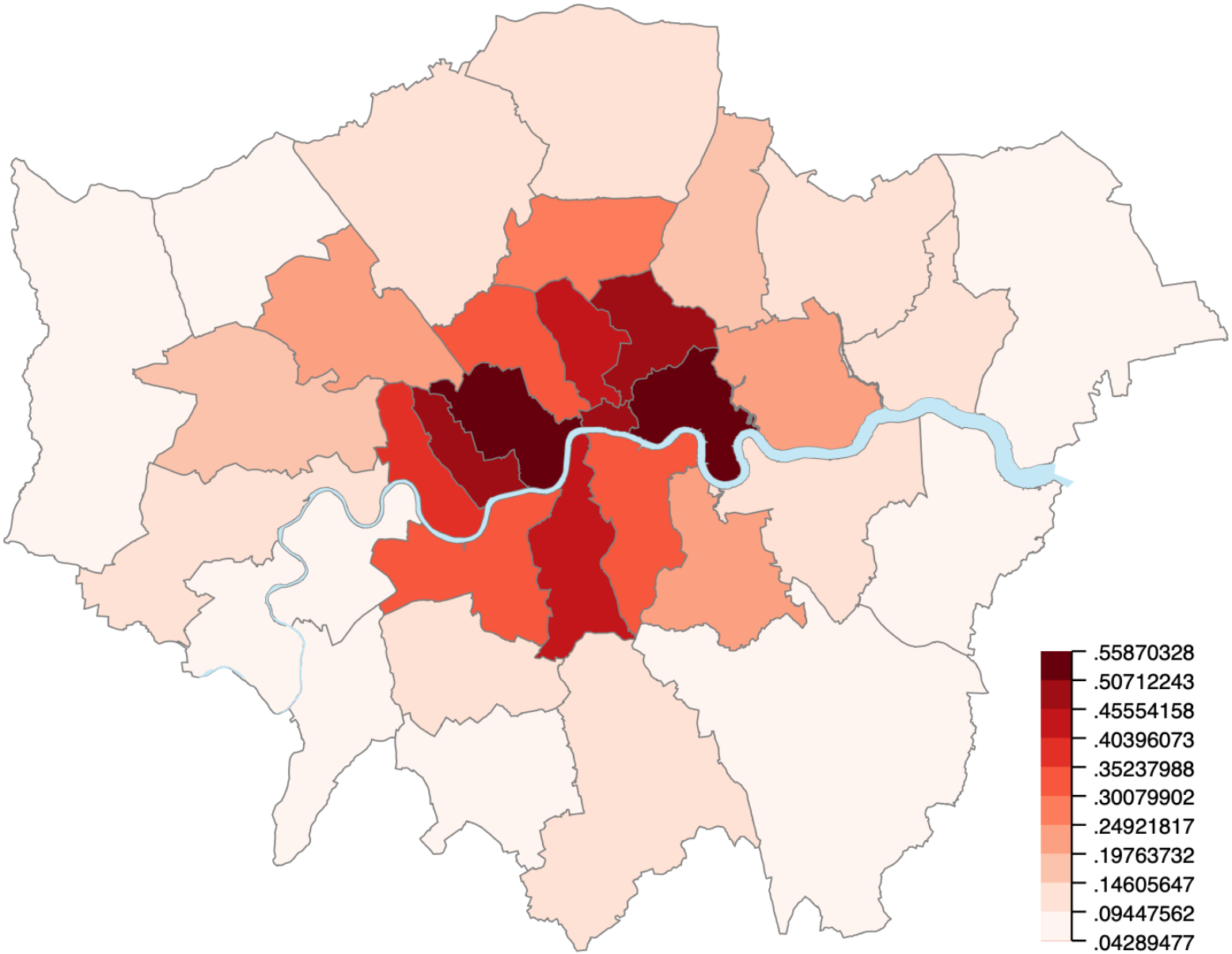
```
. frame Accidents: geoframe spjoin Borough
(plevel not set; assuming that there are no nested polygons)
(0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(77974 points not matched)
(variable _ID added to frame Accidents)
. geoplot (line Borough) (area Thames) ///
>     (point Accidents if _ID<., msymbol(p) pstyle(p2)), tight
```

Accidents

- After the spatial join we can use `geoframe contract` or `geoframe collapse` to summarize the accident data by borough.

```
. frame change Borough
. geoframe contract Accidents, id(_ID)
(variable _freq added to frame Borough)
. 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)
```

Accidents per Hectare



Conclusions

- `geoplot` provides a powerful and (relatively) easy to use toolbox for creating maps in Stata. I hope you like it.
- Installation from SSC (or from github.com/benjann/geoplot):

```
. ssc install geoplot, replace
. ssc install palettes, replace
. ssc install colrspace, replace
. ssc install moremata, replace
. ssc install geo2xy, replace // if you want to apply projections
```

- Thanks to Asjad Naqvi for extensive testing and many suggestions; also see Asjad's post on `geoplot` at medium.com/the-stata-guide.
- Some future plans
  - Better support for shapfile conversion (e.g. GeoJSON) and projections.
  - More spatial algorithms (e.g. buffering).
  - Support for bivariate maps (see `bimap` by Asjad).
  - Legend option for custom symbols.
  - Open Street Map interface.
  - ...