Drawing maps in Stata using geoplot

Ben Jann

University of Bern

2025 Stata Conference
31 July–1 August 2025
Nashville, TN

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 has its limitations.
- This is why I wrote a new command called `geoplot`.
- This is a bit like reinventing the wheel as there are many powerful tools for creating maps (in R or Python, QGIS, . . .). Nonetheless, I hope my Stata implementation will prove useful for applied researchers.

[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.

A first example

- Load some data:

```
// shapes and other data of Italian regions
geoframe create regions Italy-RegionsData.dta, id(id) ///
    coord(xcoord ycoord) shp(Italy-RegionsCoordinates.dta)
// coordinates and other data of Italian cities
geoframe create cities  Italy-Capitals.dta, coord(xcoord ycoord)
// shapes of lakes and rivers
geoframe create lakes   Italy-Lakes.dta, feature(water)
geoframe create rivers  Italy-Rivers.dta, feature(water)
```

- Draw a map:

```
geoplot ///
    (area regions, fcolor(AntiqueWhite)) ///
    (area lakes) ///
    (line rivers) ///
    (point cities if pop98>=500000, mlabel(city) pstyle(p2)) ///
    , sbar(units(km)) compass
```

A first example

geoframe - prepare the data

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

subcommand	Description
Main	
create	load data into geoframe or declare current frame as geoframe
use	load geoframe from disk
save	save geoframe to disk
describe	describe geoframe
set	update geoframe settings of current frame
get	retrieve geoframe settings of current frame
Manage shapes	
query	obtain information on units and shapes in geoframe
generate	generate helper variables related to shapes
select	select units and shapes
Manipulate shapes	
project	apply projection
rescale	rescale coordinates
clip	clip shapes using convex window
rclip	clip shapes using rectangular window
simplify	simplify (generalize) shapes
refine	add extra points to straight lines

geoframe - prepare the data

```
Generate shapes
  raster          generate raster and store in new frame
  grid            store grid lines in new frame
  tissot          store Tissot's indicatrices in new frame
  bbox            store bounding box, enclosing circle, or convex hull in new frame
  bshare          store shared borders or outlines in new frame
  symbol          generate symbol shapes and store in new frame
  symboli         symbol with immediate arguments

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

Spatial smoothing
  spsmooth        apply spatial smoothing to an attribute

Merge and append
  copy            copy and merge variables from other frame
  append          append observations from other frame
  stack           combine multiple geoframes into one
```

geoframe - prepare the data

```
Further utilities
  rename          rename geoframe
  duplicate       duplicate geoframe
  link            link shape frame to current frame
  relink          fix linkage variable after modifying data
  unlink          unlink shape frame from current frame
  clean           delete unmatched/empty shapes and units
  attach          attach attribute frame to current frame using aliases (Stata 18 required)
  detach          detach attribute frame from current frame (Stata 18 required)

Convert source
  translate       translate shapefile source to Stata format
  convert         synonym for translate
  import          translate and import shapefile source
```

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
scatter	synonym for **point**
label	single-coordinate labels
symbol	single-coordinate symbols (circles, hexagons, stars, etc.)
* **pie**	pie charts
* **bar**	stacked (or unstacked) bar charts
* **areai**	**area** with immediate arguments
* **linei**	**line** with immediate arguments
* **pointi**	**point** with immediate arguments
* **scatteri**	synonym for **pointi**
* **labeli**	**label** with immediate arguments
* **symboli**	**symbol** with immediate arguments
pcspike etc.	paired-coordinate spikes, arrows, or markers
* **pci** etc.	paired-coordinate plot with immediate arguments

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
* **gloptions(***options***)**	override options for the symbols created by **glegend()**
* **nolegend**	do not consider the layer for the default legend
Missing	
missing(*options***)**	styling of elements for which *zvar* is missing

Furthermore, various global options are available to manipulate the overall map or add elements such as legends.

global_options	Description
Main	
background[(*opts*)]	draw a background frame behind the map
grid[(*options*)]	draw grid lines on top of the map (can be repeated)
tissot[(*options*)]	draw Tissot's indicatrices on top of the map
project[(*spec*)]	apply projection (coordinates in degrees assumed)
angle(*angle*)	rotate map by *angle*
rotate(*angle*)	synonym for **angle()**
Legends	
legend[(*options*)]	add standard legend
glegend[(*options*)]	add geoplot legend (can be repeated)
slegend(*spec*)	add size legend (can be repeated)
clegend[(*options*)]	add **contour** plot legend (requires Stata 18)
sbar[(*options*)]	add scale bar
compass[(*options*)]	add compass
Zoom / Inset	
zoom(*spec*)	zoom in on specific layers (can be repeated)
inset(*spec*)	create inset containing additional map (can be repeated)
Overall appearance	
tight	adjust graph size to dimension of map
margin(*spec*)	specify (minimum) margin around map
refdim(*spec*)	select reference dimension
aspectratio(*spec*)	adjust aspect ratio of map
axes	turn display of Stata's coordinate system on
twoway_options	twoway options, other than **by()**

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" from 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.csv" from www.data.gov.uk).
- Shape file of River Thames from github.com/geotheory/londonShapefiles.

Step 2: Convert data for use in Stata

- GIS shape files are typically provided in ESRI (Environmental Systems Research Institute) format. For example, "statistical-gis-boundaries-london.zip" contains the following files.

```
. ls statistical-gis-boundaries-london/ESRI/, wide
London_Borough_Excluding_MHW.dbf
London_Borough_Excluding_MHW.GSS_CODE.atx
London_Borough_Excluding_MHW.NAME.atx
London_Borough_Excluding_MHW.prj
London_Borough_Excluding_MHW.sbn
London_Borough_Excluding_MHW.sbx
London_Borough_Excluding_MHW.shp
London_Borough_Excluding_MHW.shp.xml
London_Borough_Excluding_MHW.shx
London_Ward_CityMerged.BOROUGH.atx
London_Ward_CityMerged.cpg
London_Ward_CityMerged.dbf
London_Ward_CityMerged.GSS_CODE.atx
London_Ward_CityMerged.LB_GSS_CD.atx
London_Ward_CityMerged.prj
London_Ward_CityMerged.sbn
London_Ward_CityMerged.sbx
London_Ward_CityMerged.shp
London_Ward_CityMerged.shp.xml
London_Ward_CityMerged.shx
London_Ward.BOROUGH.atx
etc...
```

Step 2: Convert data for use in Stata

- We can use `geoframe convert` (or official Stata's `spshape2dta`) to transform ESRI shape files into to Stata format.
- For example, to translate the data on boroughs (districts), we could type:

```
. geoframe convert Borough using ///
>     statistical-gis-boundaries-london/ESRI/London_Borough_Excluding_MHW.shp ///
>     , replace
(importing shp file) (5 vars, 48,584 obs)
(importing dbf file) (7 vars, 33 obs)
(files Borough.dta and Borough_shp.dta saved)
(type geoframe create Borough to load the data)
```

- As can be seen, this creates two files, an "attribute" file containing information on the units in the data (`Borough.dta`), and a secondary file containing the shape definitions (i.e., the coordinates of the borough outlines; `Borough_shp.dta`)
- I skip translating more shape files here because `geoframe` can also import ESRI (or GeoJSON) data directly (see below).

Step 2: Convert data for use in Stata

- Accidents and well-being scores come in CSV and Excel format, respectively. For convenience, I do convert these datasets here (using Stata's `import` command) and save them in Stata format.

```
. // accidents
. import delimited 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 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

- The converted data on boroughs can be loaded into frames for use with `geoplot` using command `geoframe create`.

```
. geoframe create Borough
(creating frame Borough from Borough.dta)
(creating frame Borough_shp from Borough_shp.dta)
            Frame name: Borough [make current]
            Frame type: attribute
          Feature type: <none>
         Number of obs: 33
               Unit ID: _ID
           Coordinates: _CX _CY
    Linked shape frame: Borough_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 Borough_shp
           Frame name: Borough_shp [make current]
           Frame type: shape
         Feature type: <none>
       Number of obs: 48,584
             Unit ID: _ID
         Coordinates: _X _Y
  Within-unit sort ID: shape_order
```

Step 3: Load data into frames using geoframe

- I now also load the data on wards (urban quarters) by direct import of the ESRI source from the zip file.

```
. geoframe create Ward using ///
>     statistical-gis-boundaries-london.zip/ESRI/London_Ward_CityMerged.shp
(importing shp file) (5 vars, 158,520 obs)
(importing dbf file) (7 vars, 625 obs)
(creating frame Ward)
(creating frame Ward_shp)
            Frame name: Ward [make current]
            Frame type: attribute
          Feature type: <none>
         Number of obs: 625
               Unit ID: _ID
           Coordinates: _CX _CY
   Linked shape frame: Ward_shp
```

Step 3: Load data into frames using geoframe

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

```
. frame Borough: describe
Contains data from Borough.dta
 Observations:            33
    Variables:            10                  21 May 2025 18:07
```

Variable name	Storage type	Display format	Value label	Variable label
_ID	byte	%12.0g		Spatial-unit ID
_CX	double	%10.0g		X coordinate of centroid
_CY	double	%10.0g		Y coordinate of centroid
NAME	str22	%22s		NAME
GSS_CODE	str9	%9s		GSS_CODE
HECTARES	double	%12.3f		HECTARES
NONLD_AREA	double	%12.3f		NONLD_AREA
ONS_INNER	str1	%9s		ONS_INNER
SUB_2009	str1	%9s		SUB_2009
SUB_2006	str1	%9s		SUB_2006

```
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
(creating frame Wellbeing from Wellbeing.dta)
(frame Wellbeing made current)
. rename Crimerate2013 Crimerate
. rename AW Jobless
. frame Ward: geoframe copy Wellbeing Crimerate Jobless, id(GSS_CODE)
(all units in frame Ward matched)
(2 variables copied from frame Wellbeing)
. frame Borough: geoframe copy Wellbeing Crimerate Jobless, id(GSS_CODE)
(all units in frame Borough matched)
(2 variables copied from frame Wellbeing)
. frame Ward: describe
Contains data
 Observations:           625
    Variables:            12
```

Variable name	Storage type	Display format	Value label	Variable label
_ID	int	%12.0g		Spatial-unit ID
_CX	double	%10.0g		X coordinate of centroid
_CY	double	%10.0g		Y coordinate of 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

Step 3: Load data into frames using geoframe

- I now also load the accidents data (which is only an attribute file without shape file) . . .

```
. geoframe create Accidents
(creating frame Accidents from Accidents.dta)
            Frame name: Accidents [make current]
            Frame type: attribute
          Feature type: <none>
         Number of obs: 101,087
               Unit ID: <none>
           Coordinates: _X _Y
    Linked shape frame: <none>
```

- . . . import the coordinates of the SIL points . . .

```
. geoframe create SIL lp-consultation-oct-2009-sil-points-shp.zip
(translating lp-consultation-oct-2009-sil-points-shp.zip/lp-consultation-oct-2009-sil
> -points.shp)
(importing shp file) (4 vars, 59 obs)
(importing dbf file) (7 vars, 59 obs)
(creating frame SIL)
(creating frame SIL_shp)
            Frame name: SIL [make current]
            Frame type: attribute
          Feature type: <none>
         Number of obs: 59
               Unit ID: _ID
           Coordinates: _CX _CY
    Linked shape frame: SIL_shp
. frame SIL: encode SES_Type, generate(Type)
```

Step 3: Load data into frames using geoframe

- . . . and the River Thames.

```
. geoframe create Thames londonShapefiles-master.zip/inst/external/river_thames ///
>     , feature(water)
(importing shp file) (5 vars, 3,017 obs)
(importing dbf file) (4 vars, 1 obs)
(creating frame Thames)
(creating frame Thames_shp)
            Frame name: Thames [make current]
            Frame type: attribute
          Feature type: water
         Number of obs: 1
               Unit ID: _ID
           Coordinates: _CX _CY
  Linked shape frame: Thames_shp
```

- Option `feature(water)` was specified to declare the type of feature included in the frame; this information will be picked up by `geoplot`.

Step 3: Load data into frames using geoframe

- Here is an overview of the frames that are now in memory.

```
. frame dir
* Accidents     101087 x 3; Accidents.dta
* Borough       33 x 12; Borough.dta
* Borough_shp   48584 x 5; Borough_shp.dta
* SIL           59 x 11
* SIL_shp       59 x 4
* Thames        1 x 7
* Thames_shp    3017 x 5
* Ward          625 x 12
* Ward_shp      158520 x 5
* Wellbeing     659 x 64; Wellbeing.dta
  default       0 x 0
Note: Frames marked with * contain unsaved data.
```

- We can now use these frames in `geoplot` to create maps.

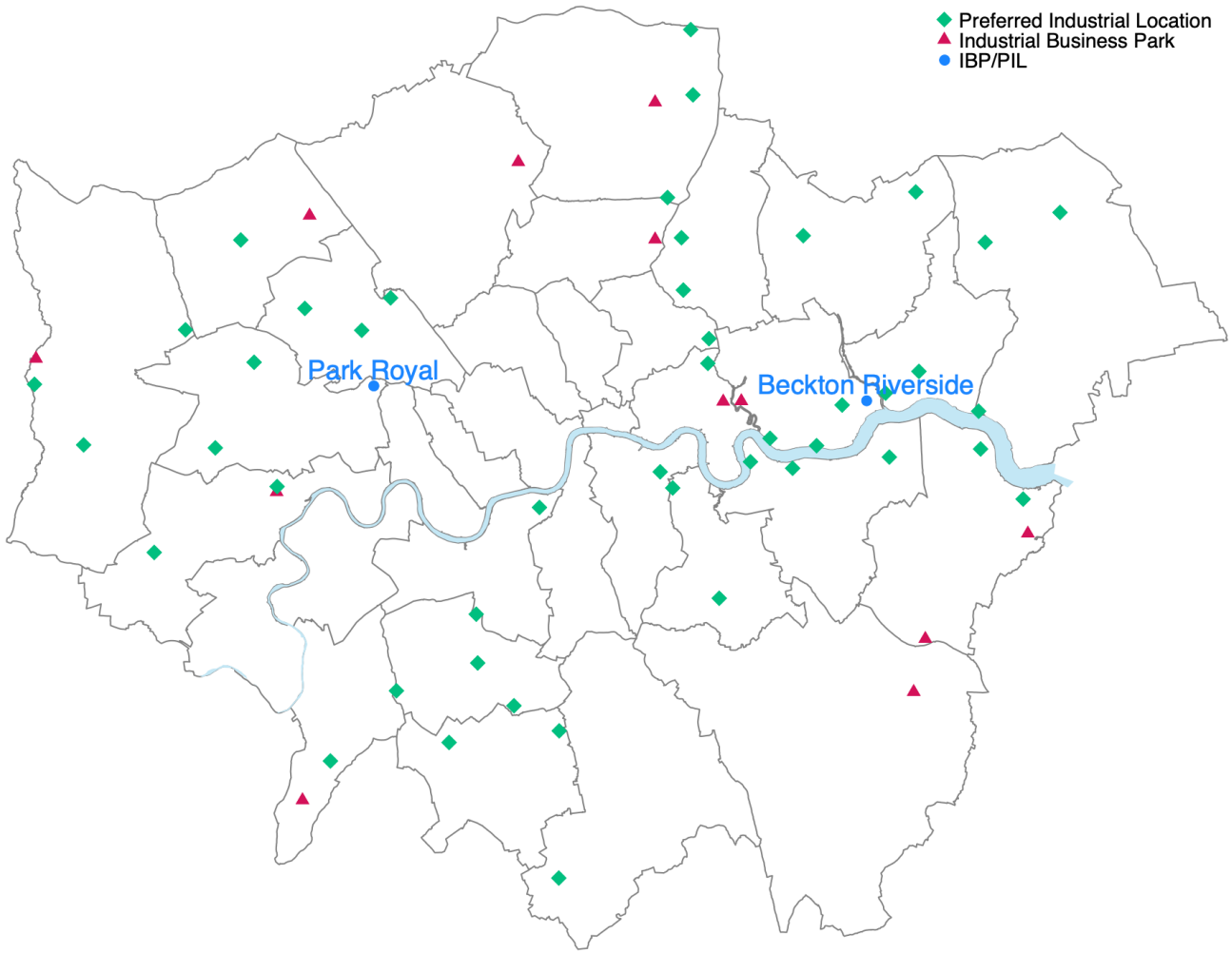
Step 4: Draw a map using geoplot

- Boroughs, wards, and river Thames.

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

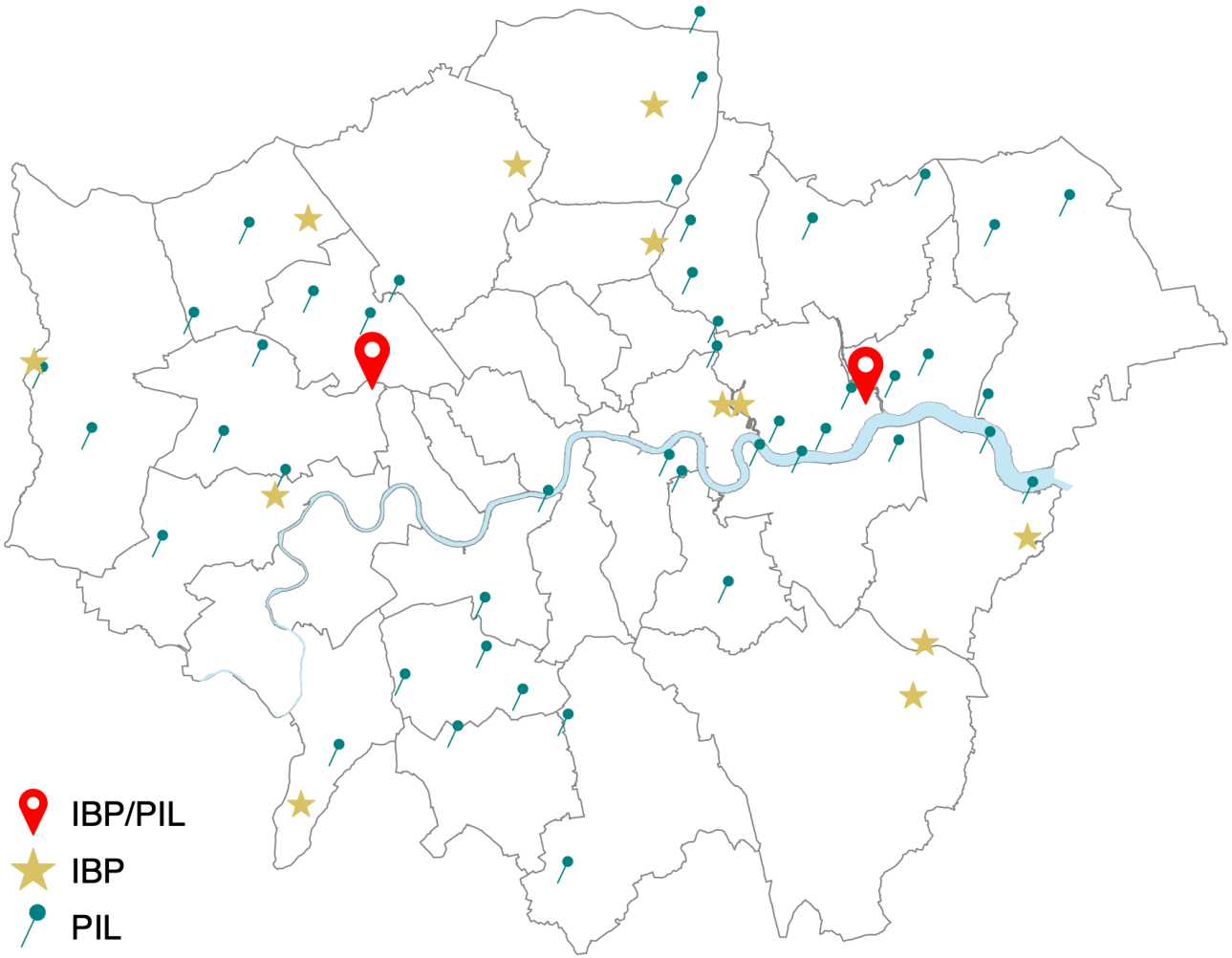
Add some points of interest . . .

```
. geoplot (line Borough) (area Thames) (point SIL i.Type, ms(o t d)) ///
>     (label SIL Location i.Type if Type==1, pos(12))
```



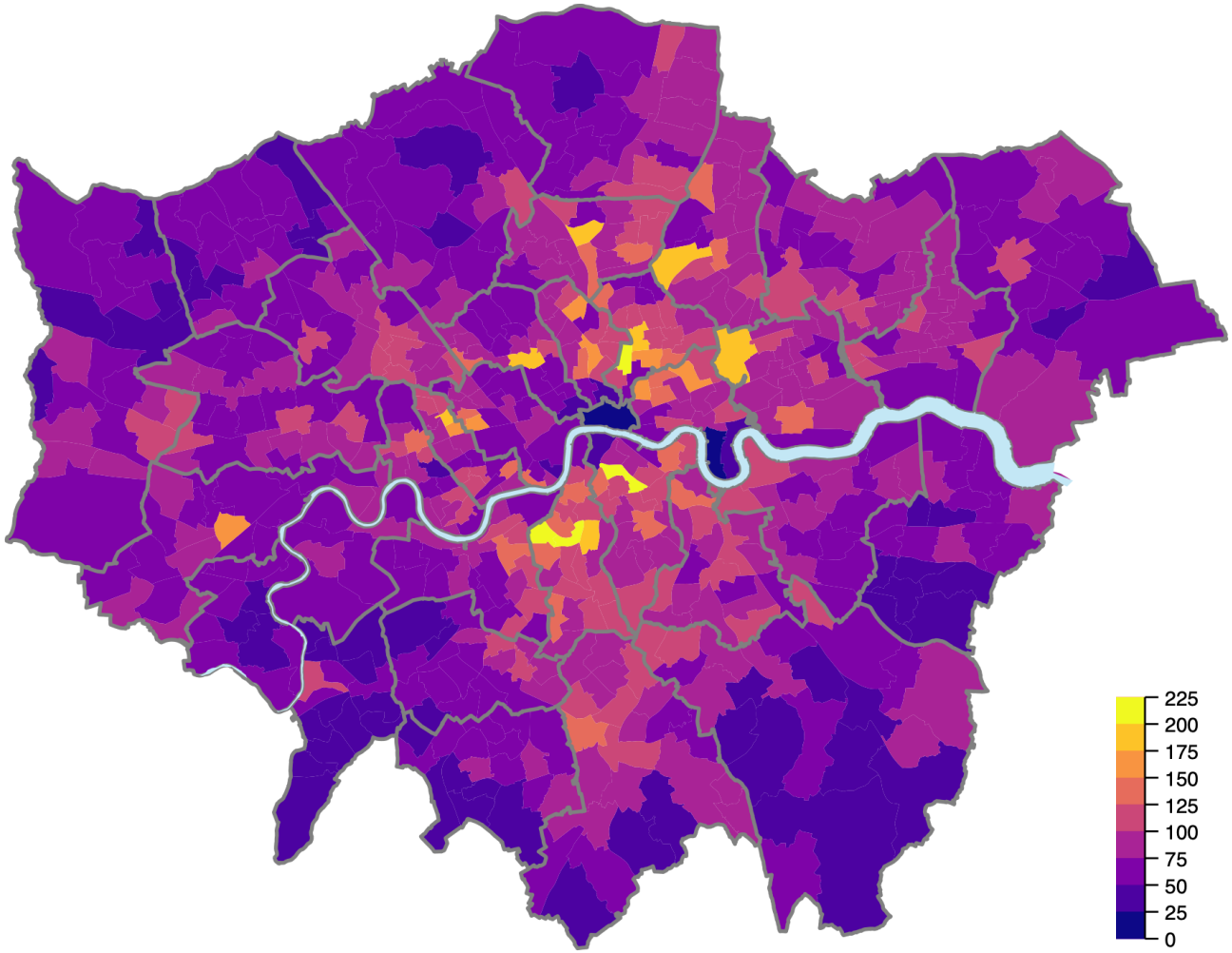
. . . with custom symbols

```
. geoplot (line Borough) (area Thames) ///
>     (symbol SIL if Type==3, label(PIL)     shape(pin) angle(-25) color(Teal)) ///
>     (symbol SIL if Type==2, label(IBP)     shape(pentagram) color(sand)) ///
>     (symbol SIL if Type==1, label(IBP/PIL) shape(pin2) color(red) size(*2)), ///
>     glegend(layout(5 4 3) symsize(6) tsize(medsmall) pos(sw))
```



Add color depending on attribute (choropleth map)

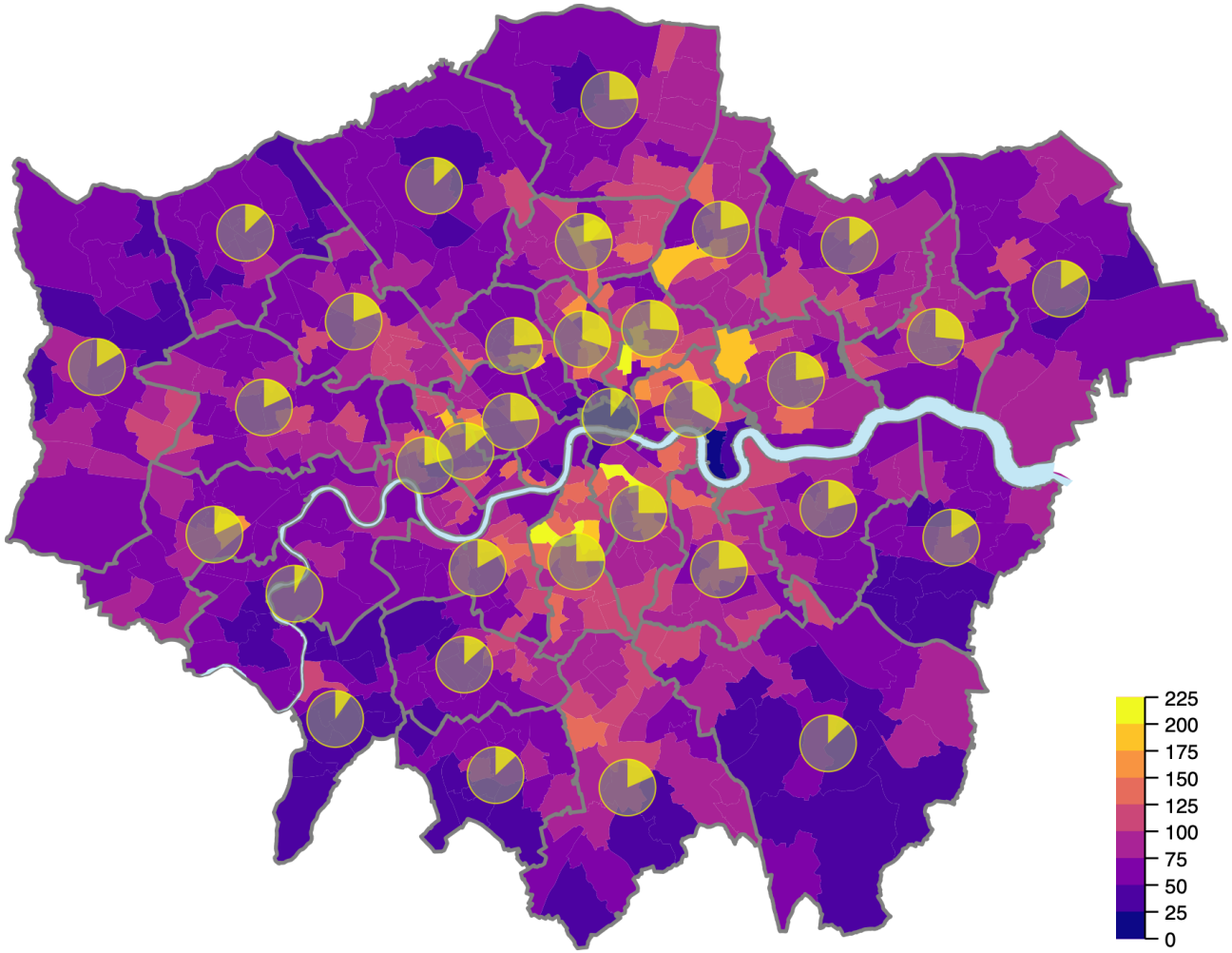
```
. geoplot (area Ward Crimerate, color(plasma) cuts(0(25)225)) ///
>     (line Borough, lwidth(.4)) (area Thames), clegend(position(se))
```



(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)), clegend(position(se))
```

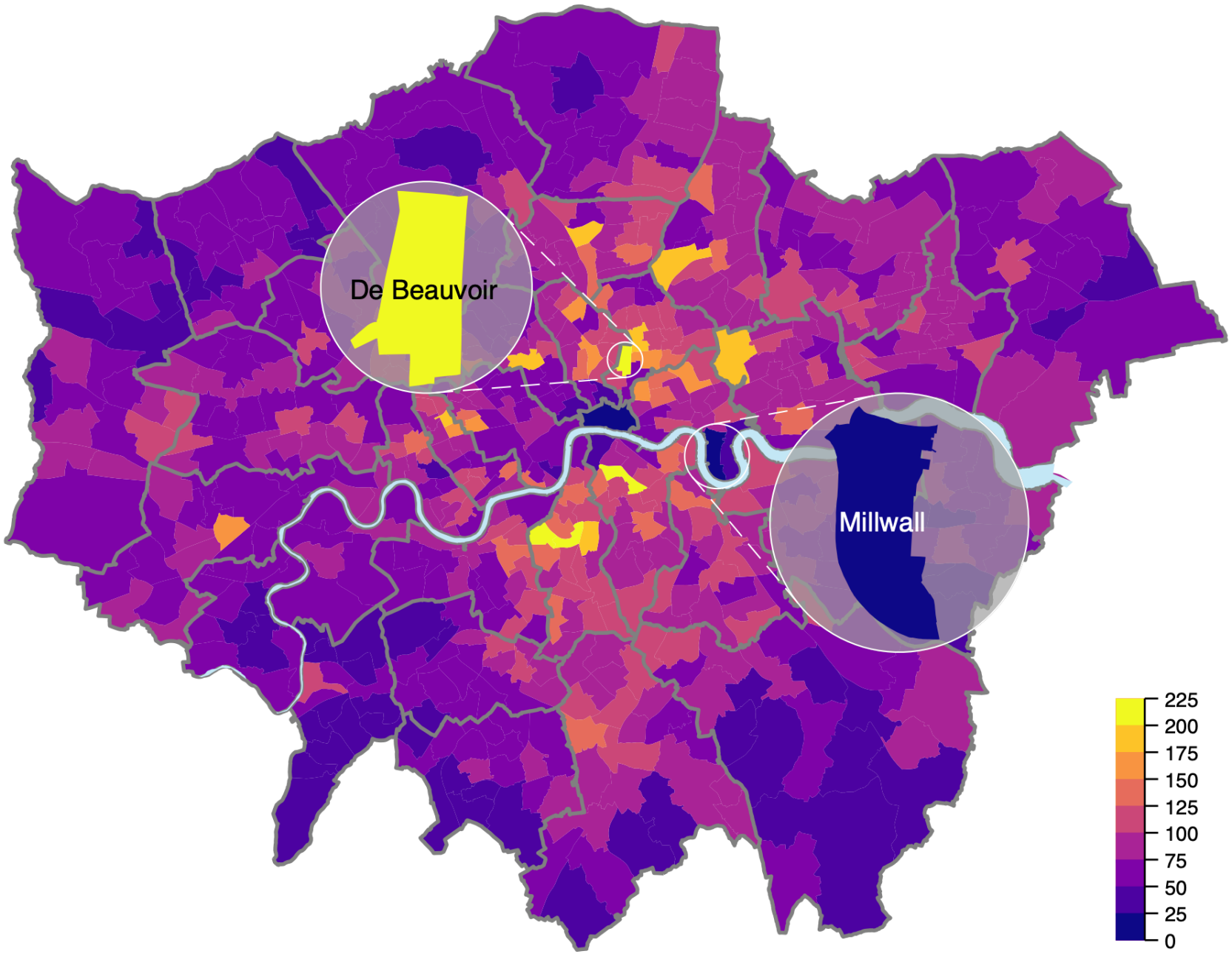


(percentage of dependent children in jobless households 2013)

Zoom in on min and max

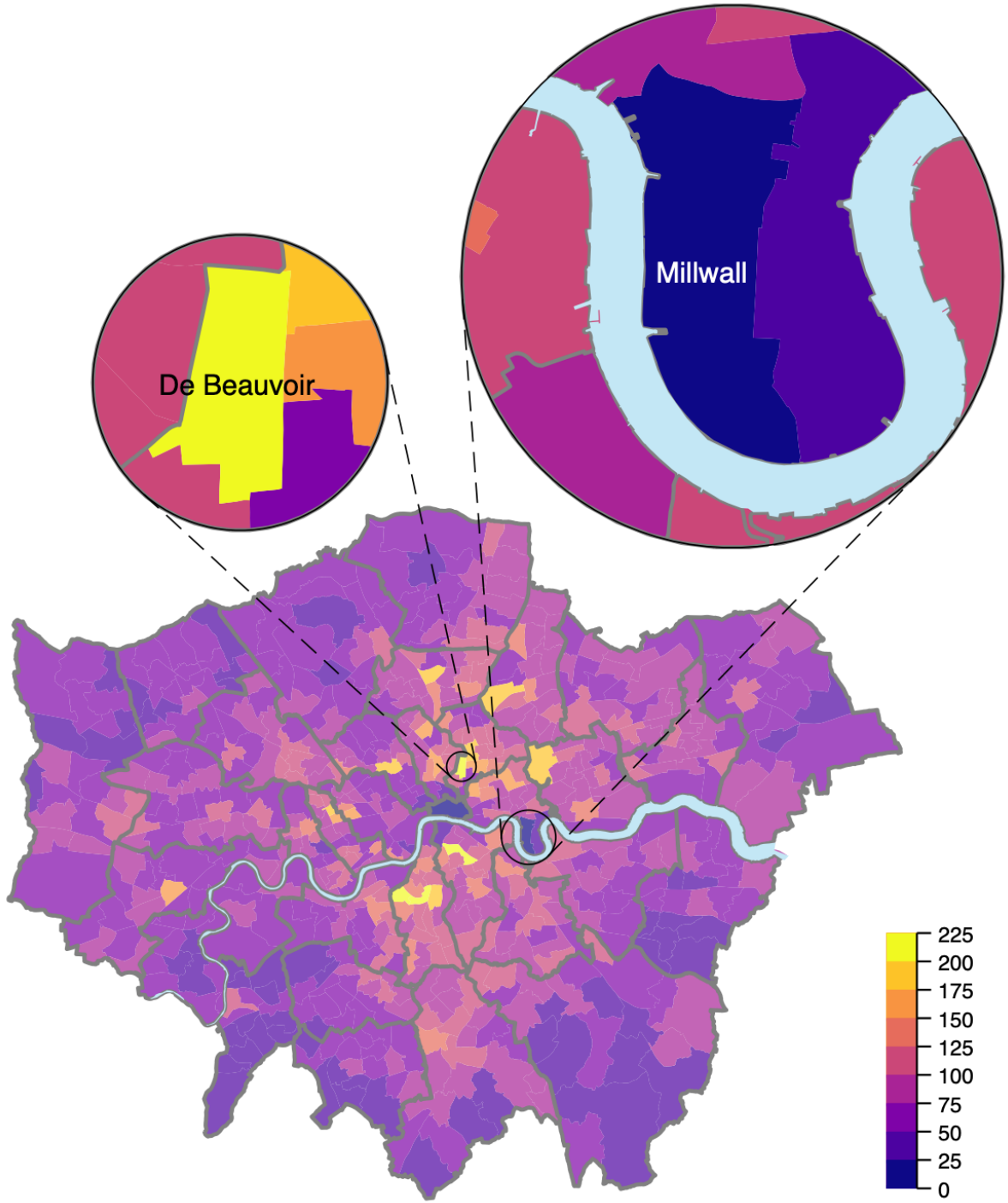
```
. frame change Ward
. su Crimerate, meanonly
. su _ID if inlist(Crimerate,r(min),r(max)), meanonly
. local min = r(min)
. local max = r(max)
. geoplot ///
>     (area Ward Crimerate, cuts(0(25)225) col(plasma)) ///
>     (line Borough, lwidth(.4)) ///
>     (area Thames) ///
>     (area Ward Crimerate,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



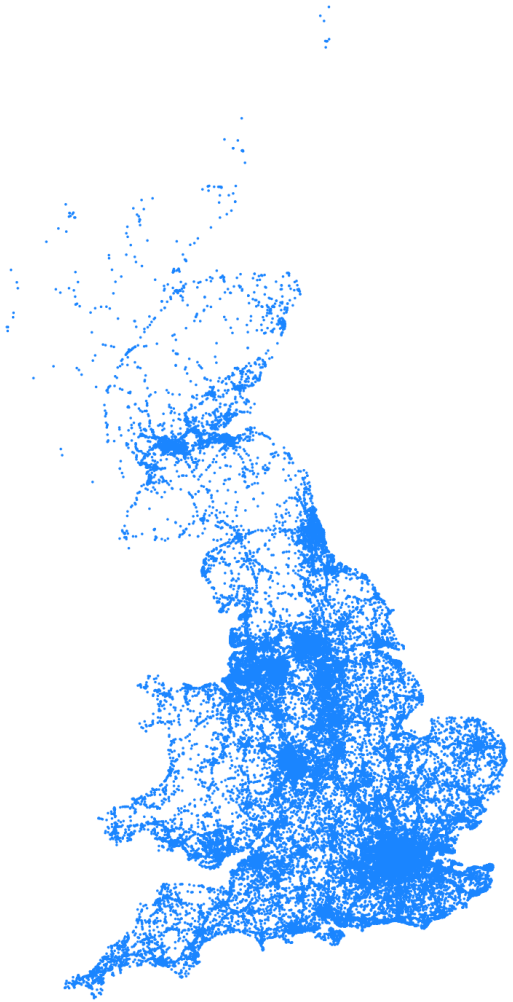
Zoom with clipped shapes

```
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

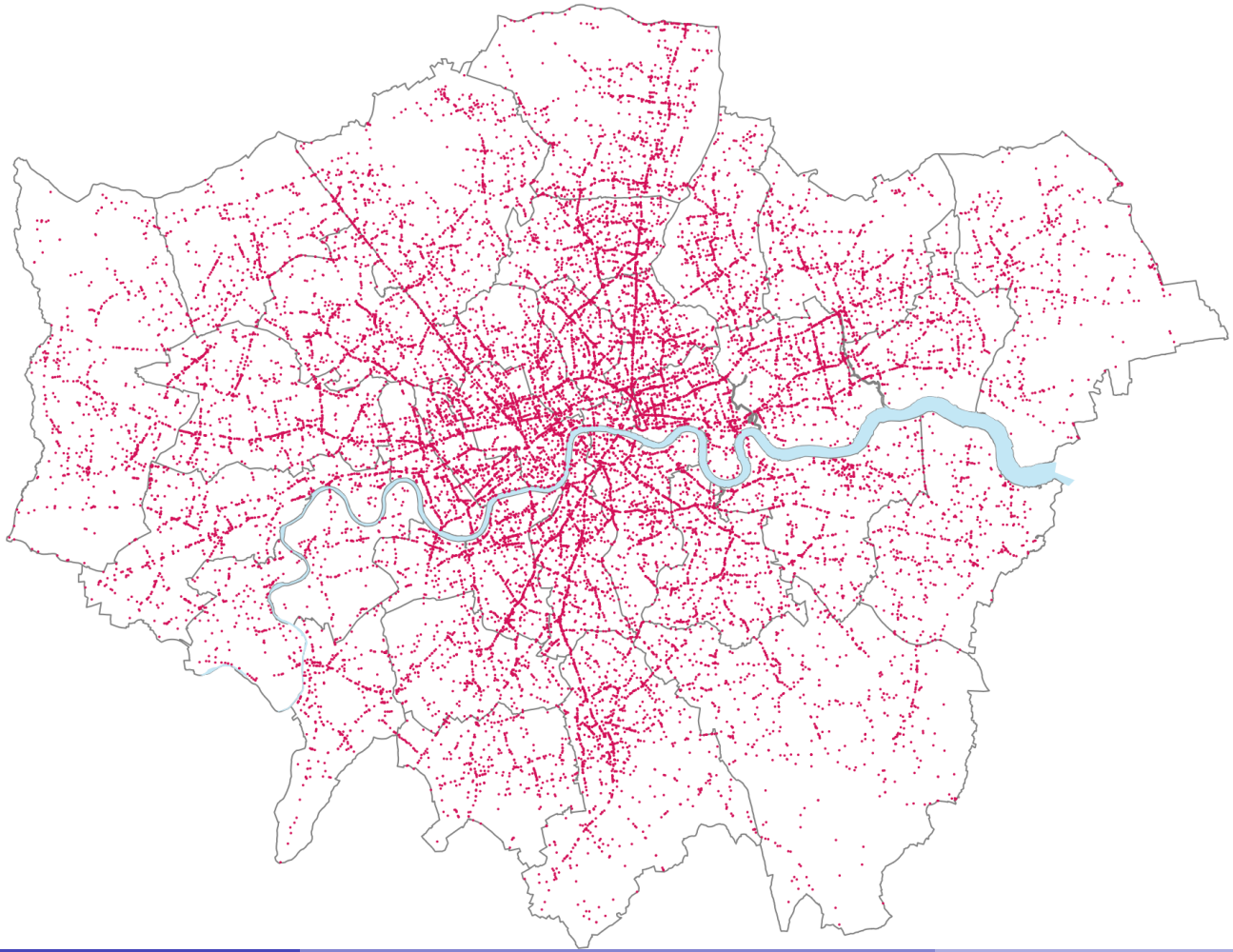
```
. geoplot (point Accidents, msymbol(p))
```



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 items)
(0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(77974 points not matched)
(variable _ID added to frame Accidents)
(data in frame Accidents sorted by _ID)
. geoplot (line Borough) (area Thames) ///
>     (point Accidents if _ID<., msymbol(p) pstyle(p2)), tight
```



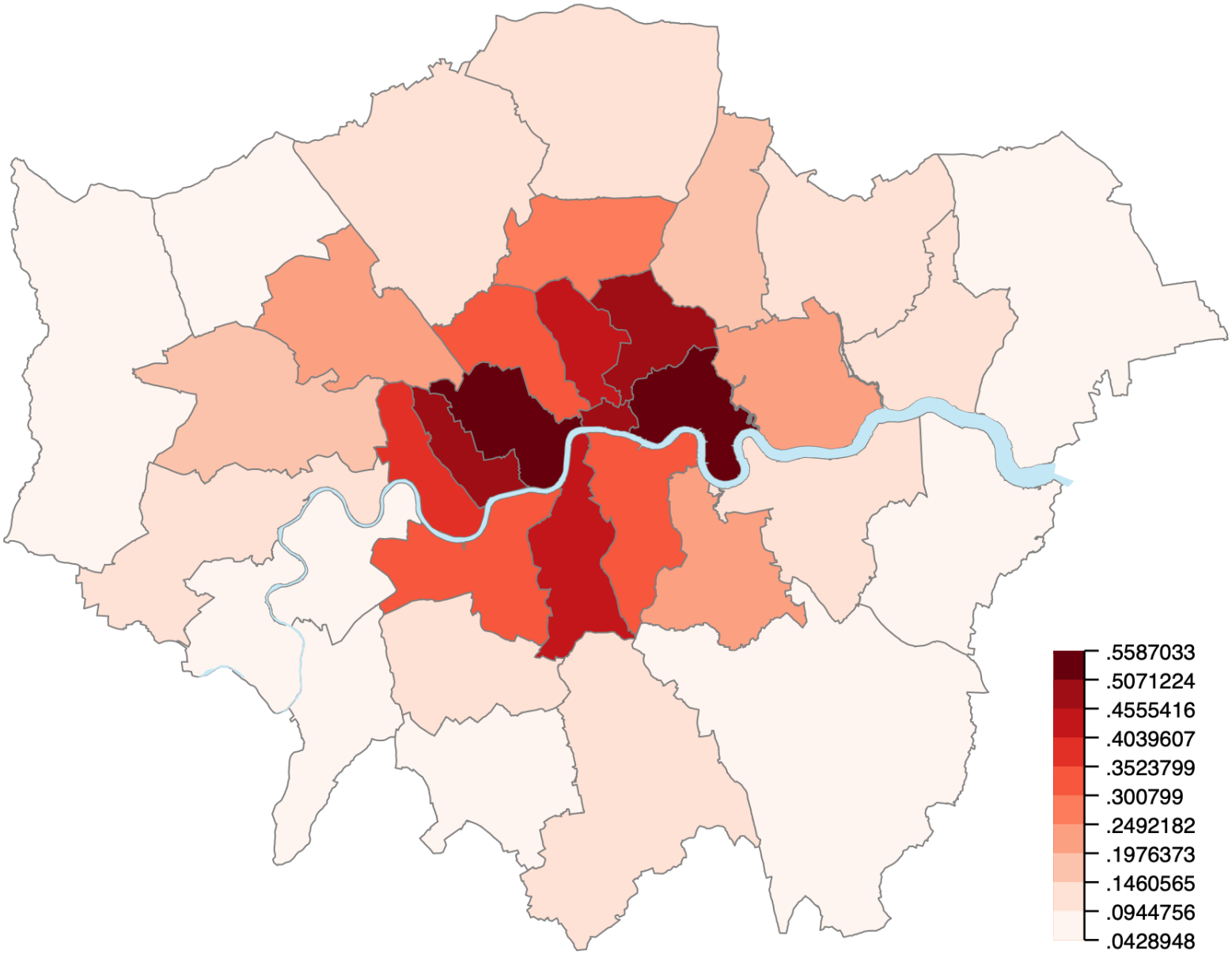
Accidents

- 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)
```

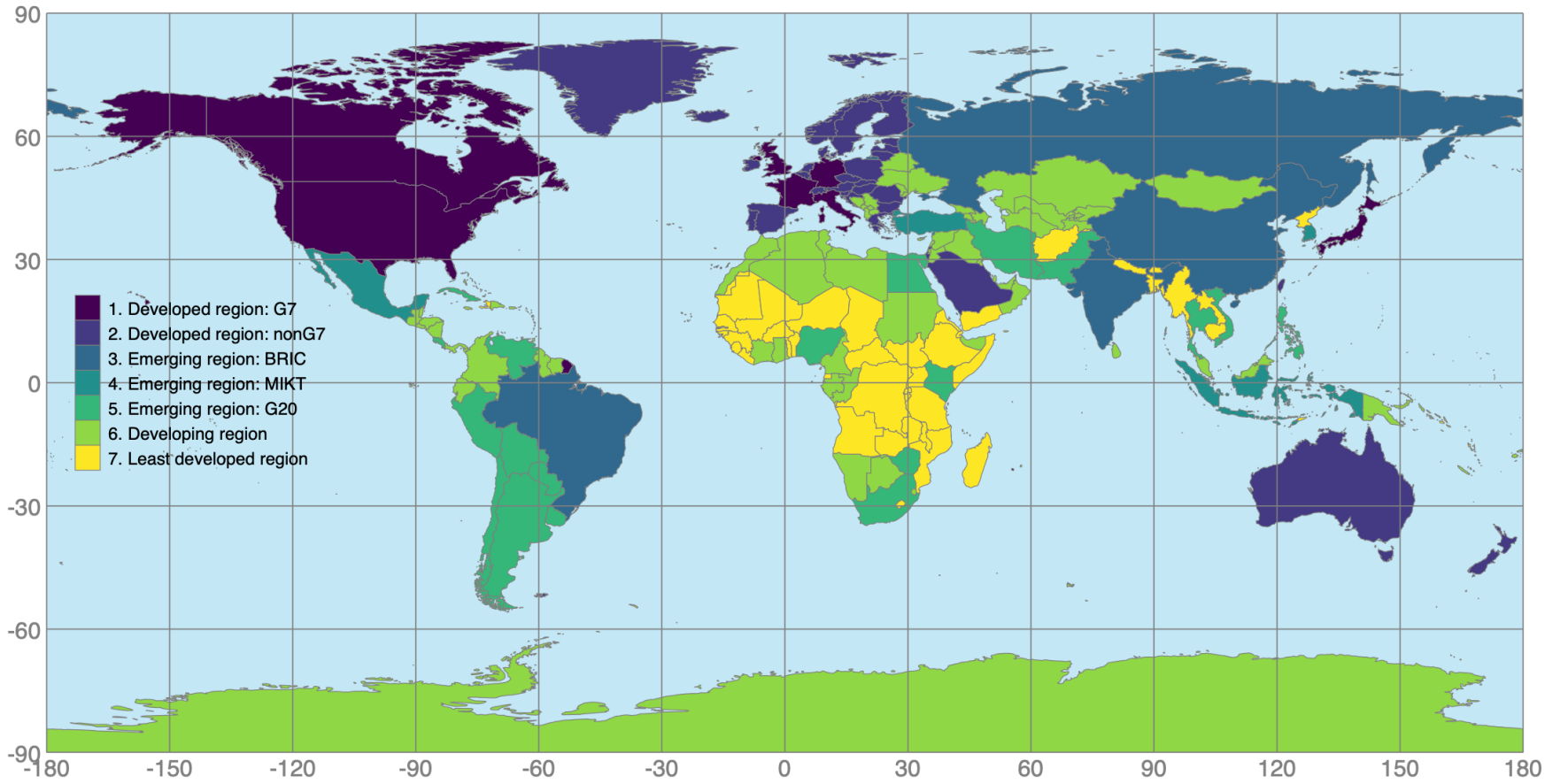
(Note that `geoframe contract` and `geoframe collapse` will automatically perform a spatial join before summarizing if option `id()` is omitted.)

Accidents per Hectare



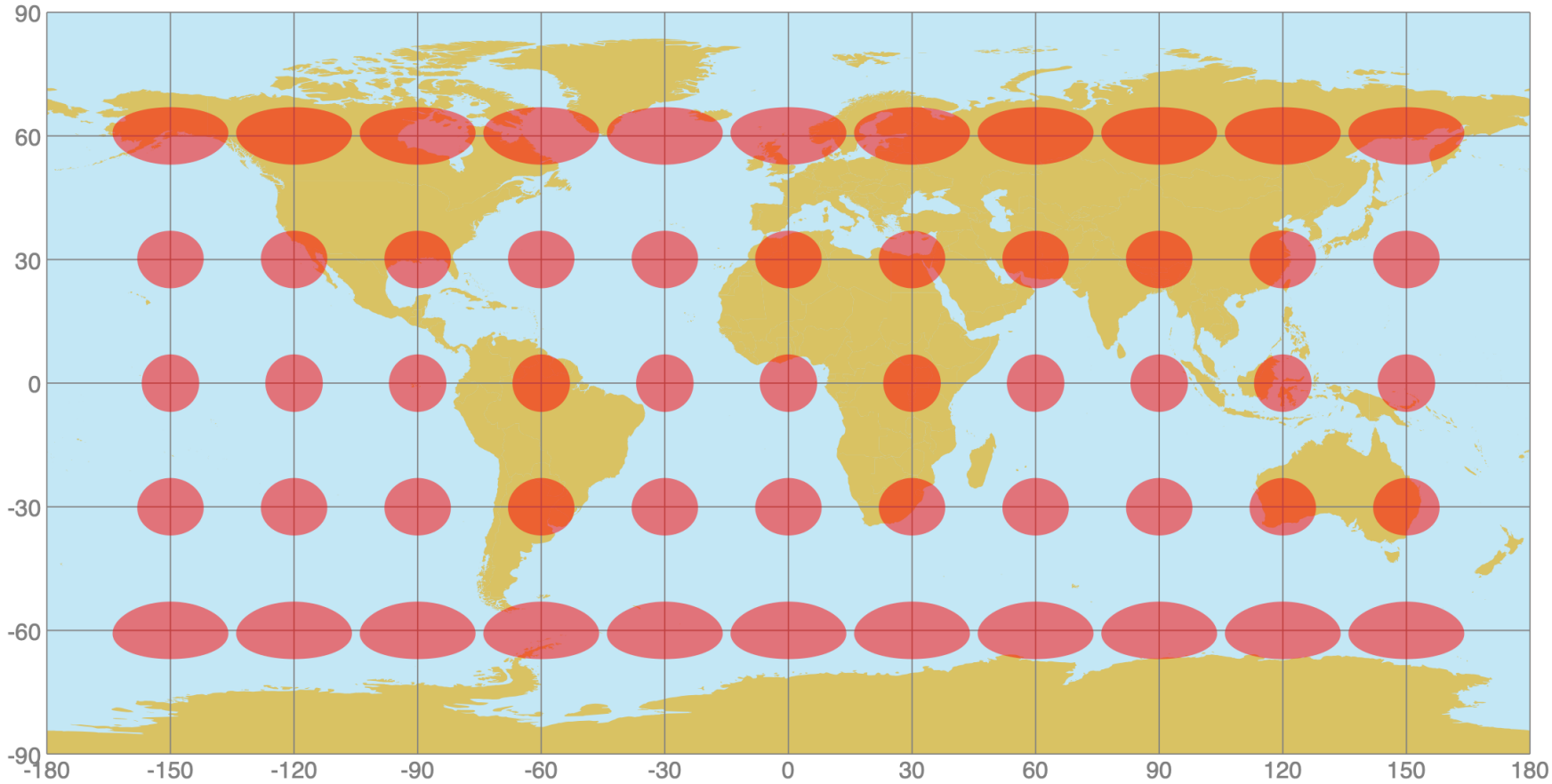
Raw data (longitude and latitude in degrees)

```
geoframe create world ne_50m_admin_0_countries.zip // (from www.naturalearthdata.com)
geoplot (area world ECONOMY, color(viridis) lc(gray) lw(.1)), tight ///
    background(water) grid(labels) legend(position(w) bm(vlarge))
```



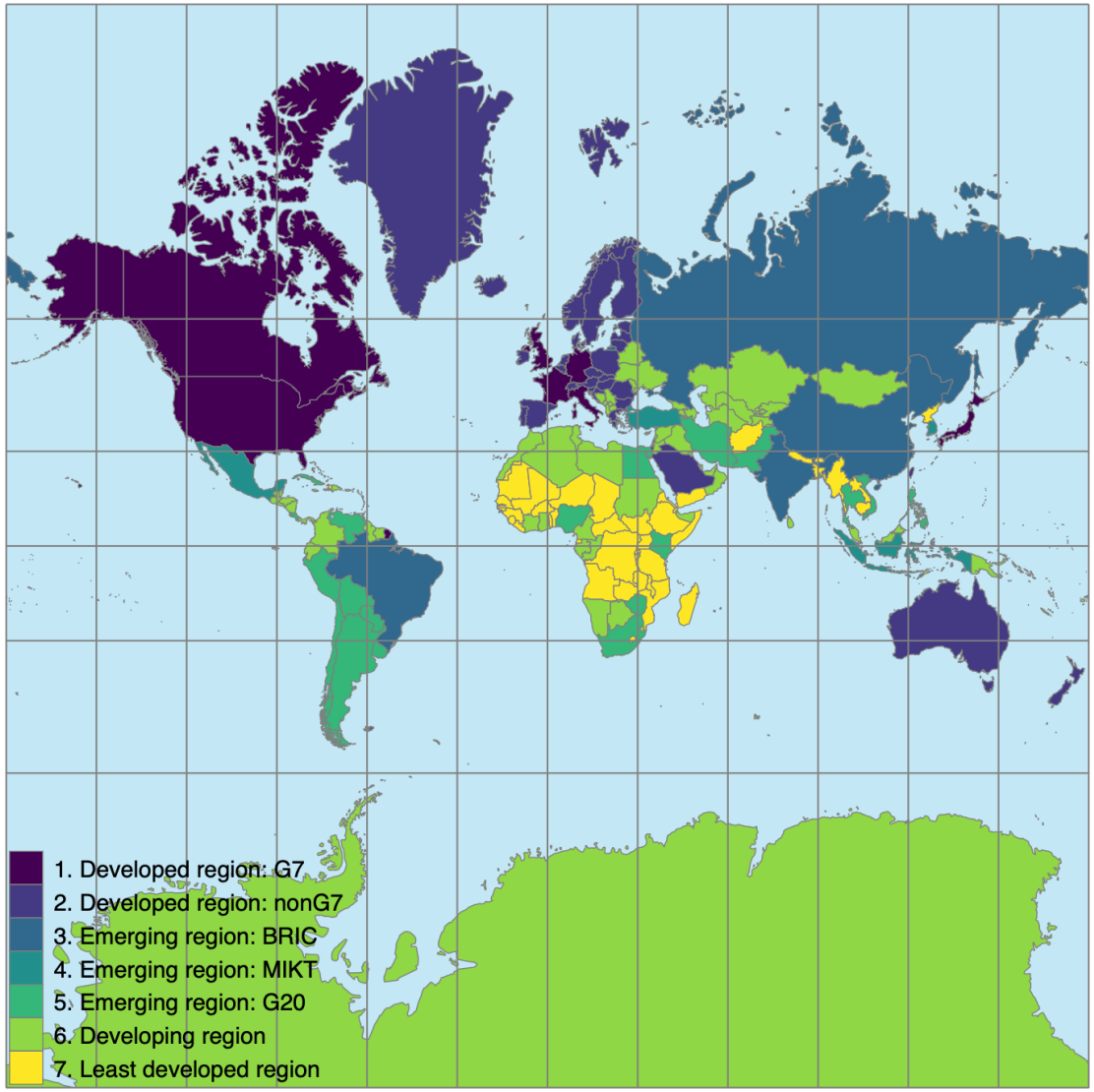
. . . with Tissot's indicatrices illustrating local distortions

```
geoplot (area world, col(sand)), tight ///
    background(water) grid(labels) ///
    tissot(col(red%50))
```



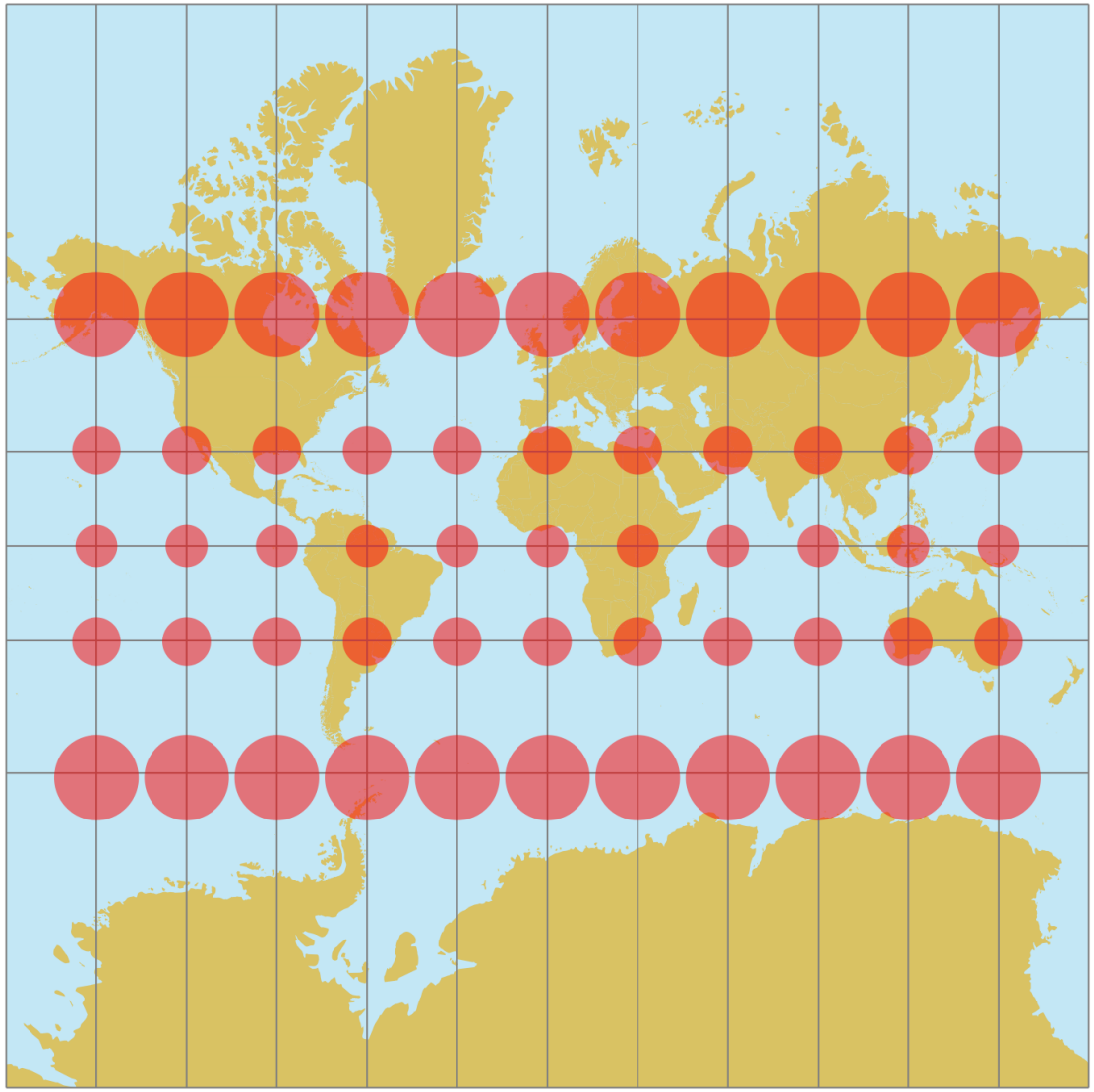
Mercator projection (used, e.g., by Google maps) (cylindrical)

```
geoplot (area world ECONOMY, color(viridis) lc(gray) lw(.1)), tight ///
    background(water) grid legend(position(sw)) project
```



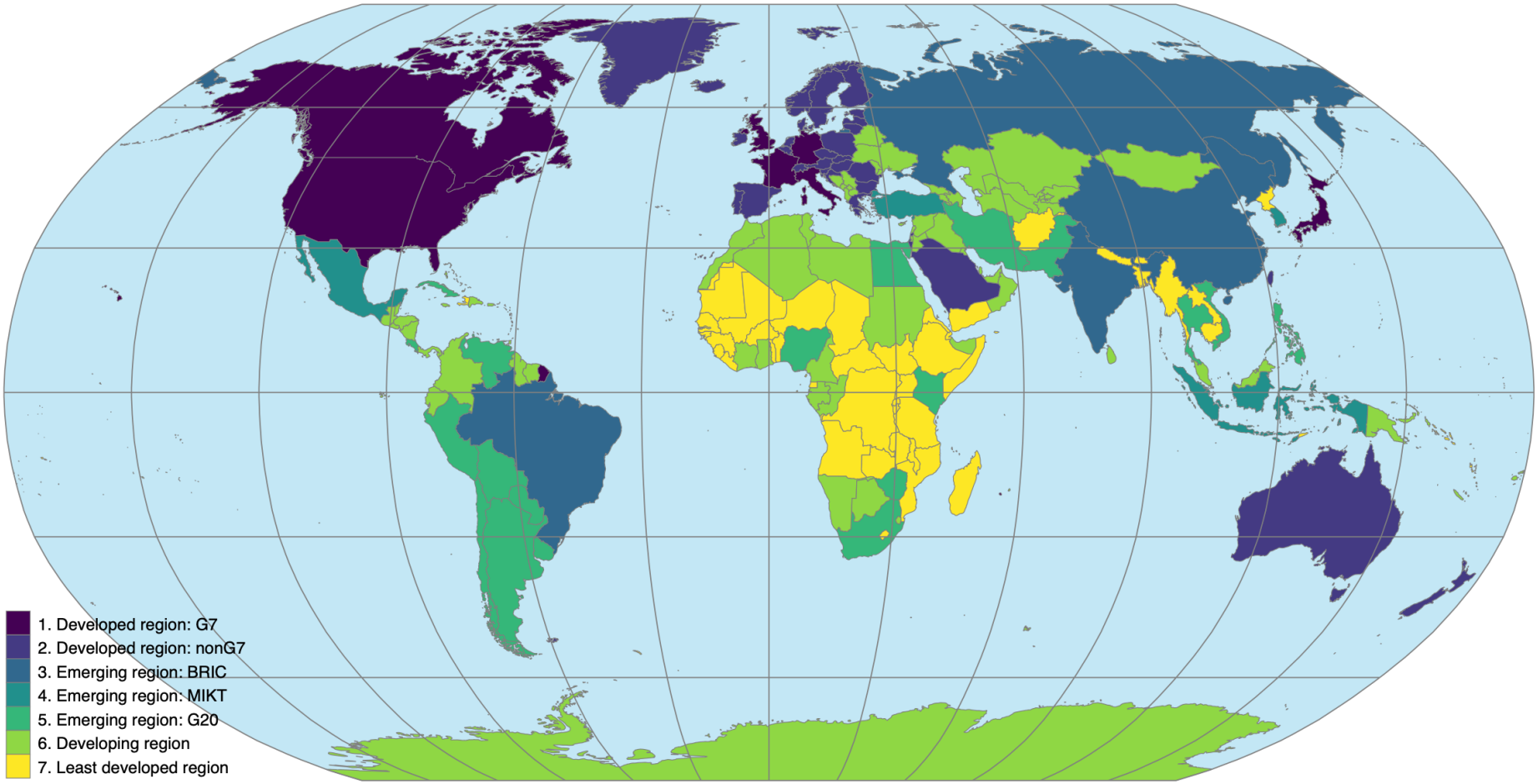
. . . with Tissot's indicatrices

```
geoplot (area world, col(sand)), tight ///
    background(water) grid project tissot(col(red%50))
```



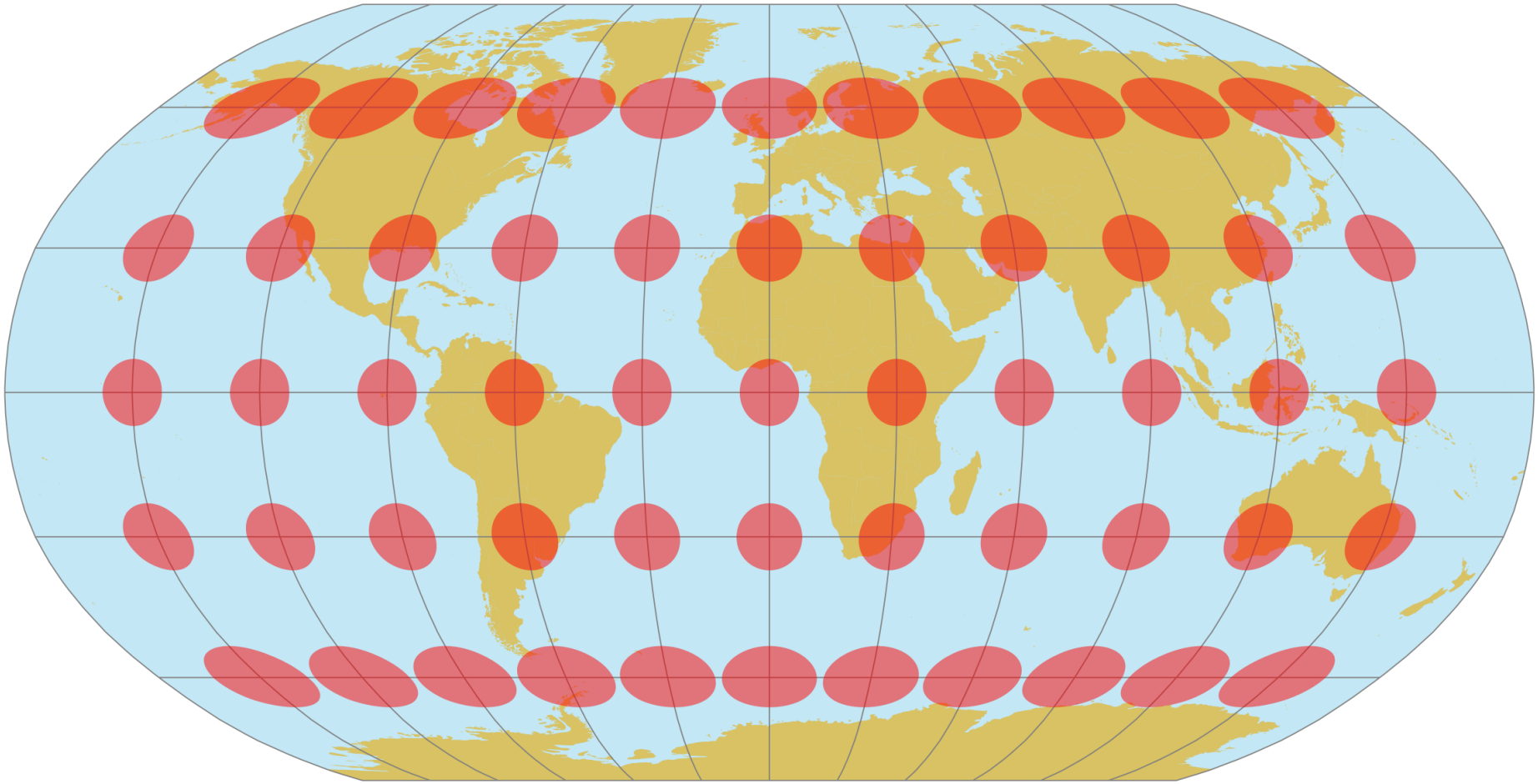
Robinson projection (pseudocylindrical)

```
geoplot (area world ECONOMY, color(viridis) lc(gray) lw(.1)), tight ///
    background(water) grid legend(position(sw)) project(robinson)
```



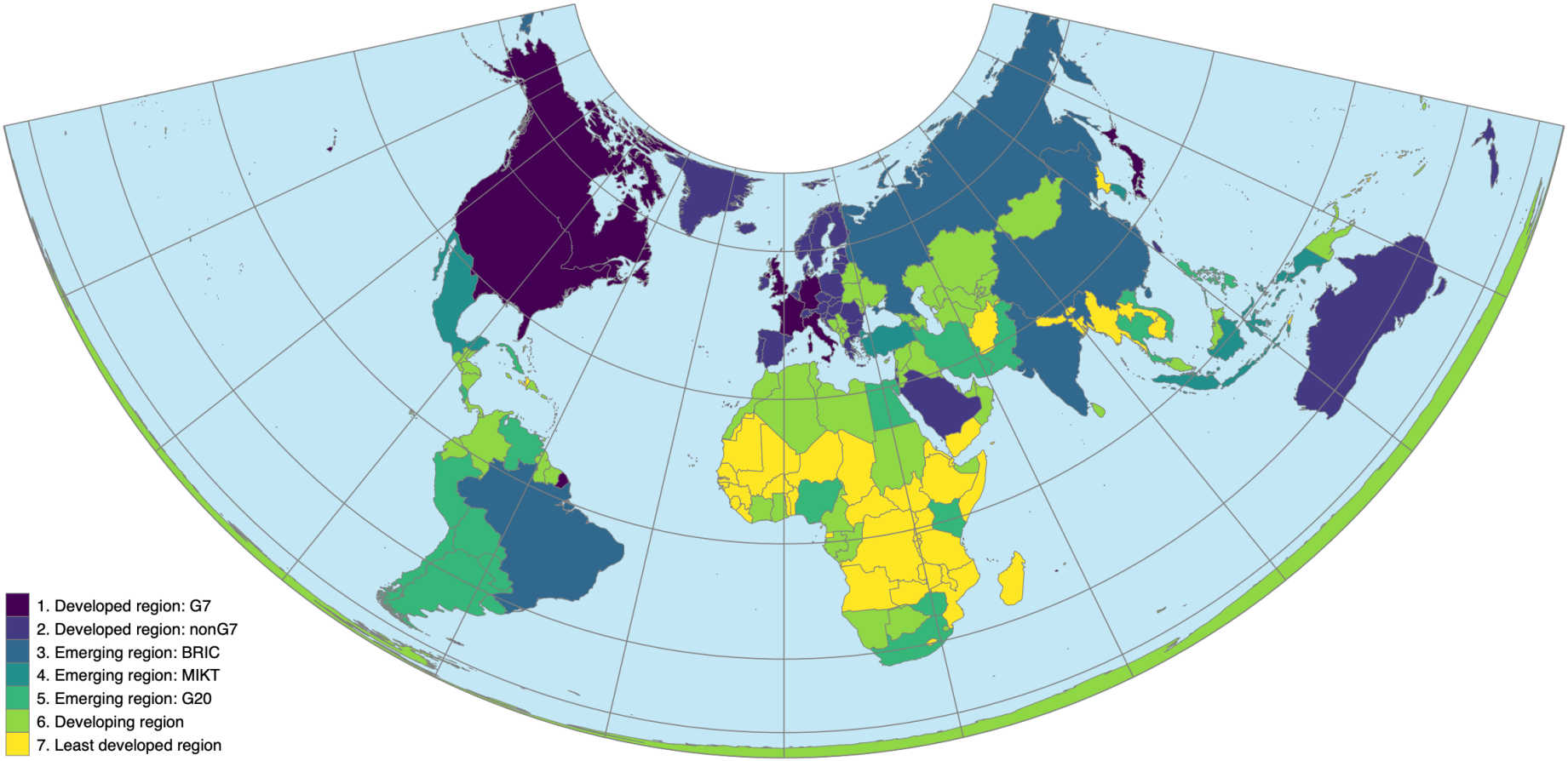
. . . with Tissot's indicatrices

```
geoplot (area world, col(sand)), tight ///
    background(water) grid project(robinson) tissot(col(red%50))
```



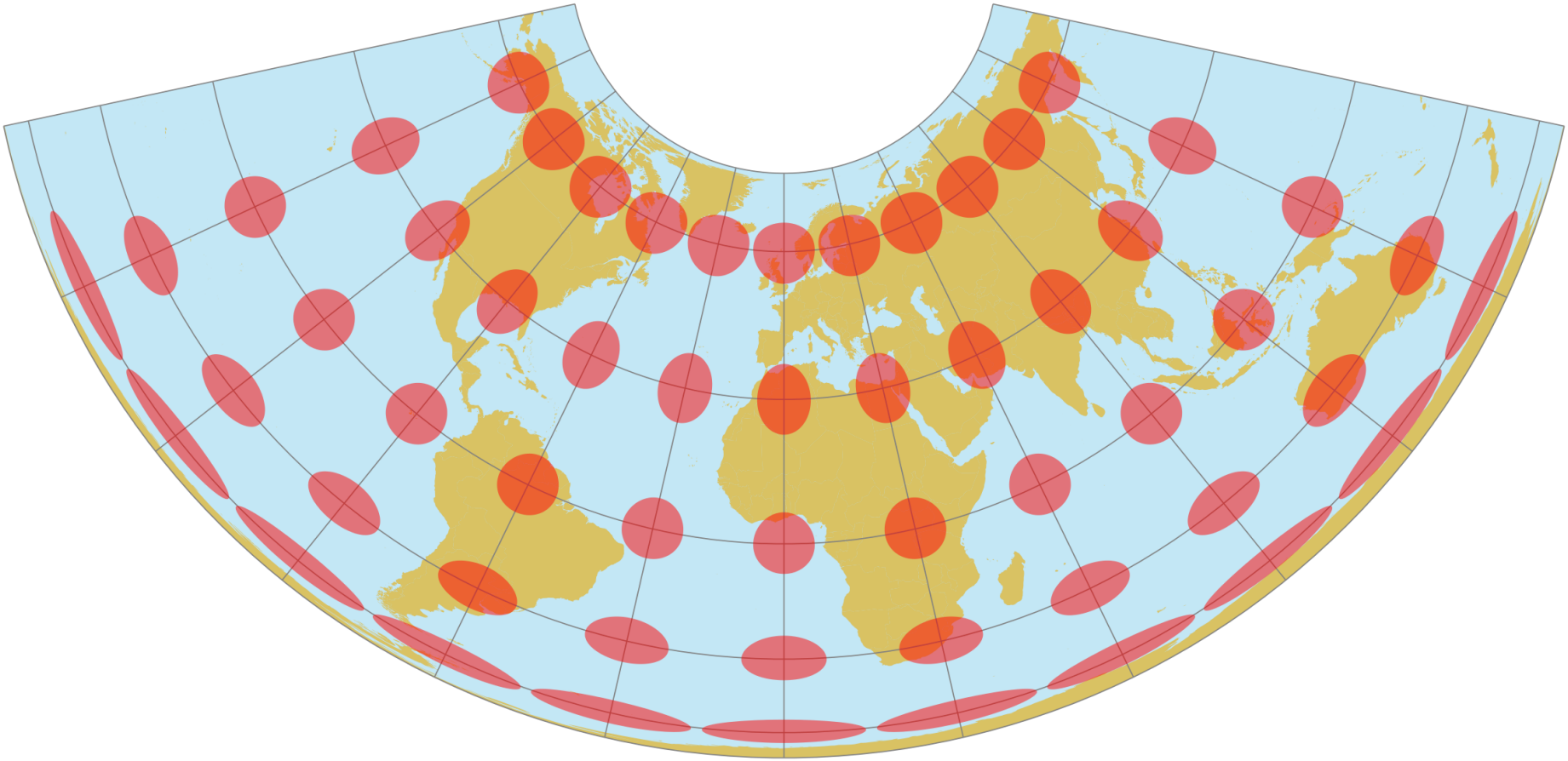
Albers projection (conic)

```
geoplot (area world ECONOMY, color(viridis) lc(gray) lw(.1)), tight ///
    background(water) grid legend(position(sw)) project(albers)
```



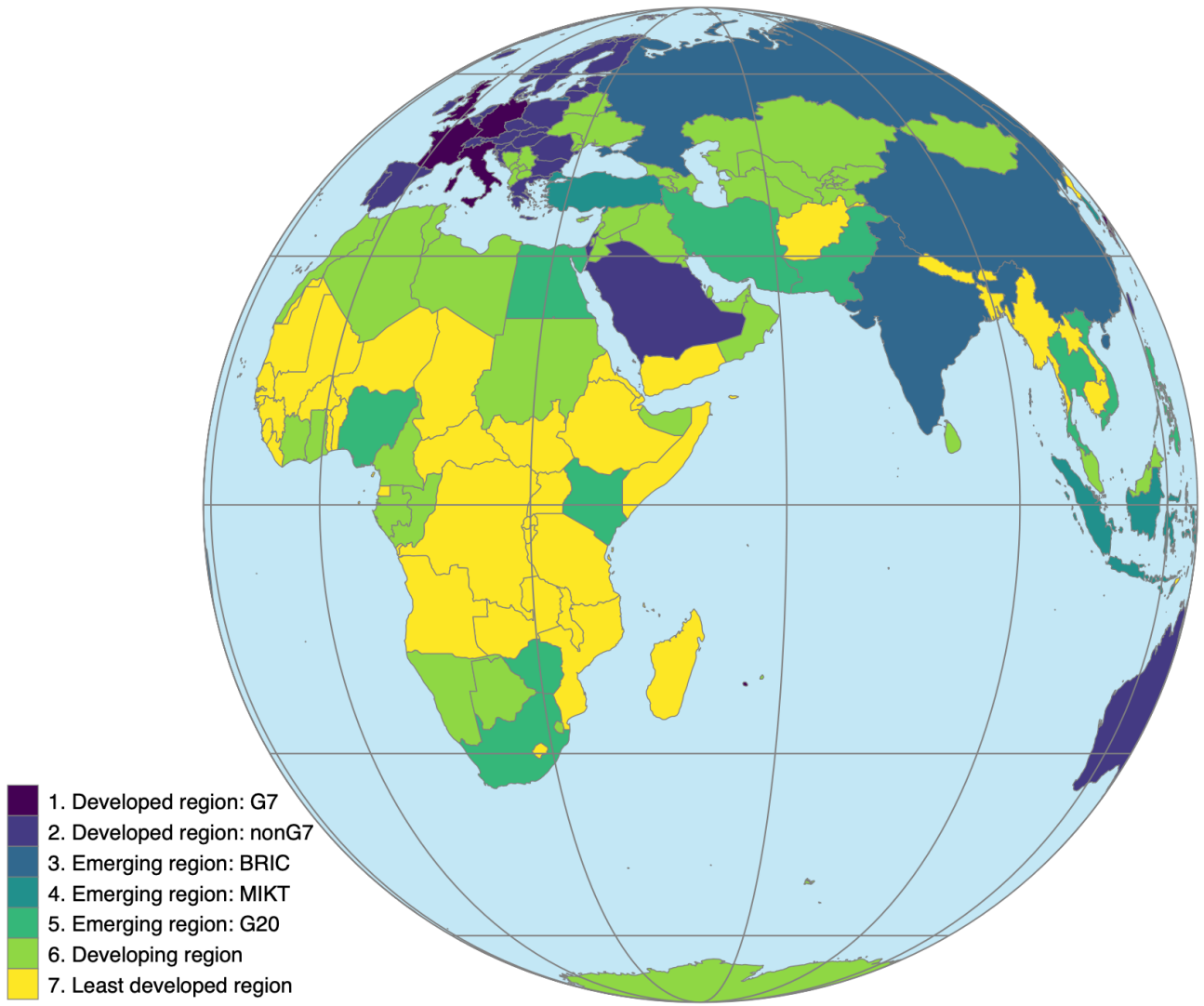
. . . with Tissot's indicatrices

```
geoplot (area world, col(sand)), tight ///
    background(water) grid project(albers) tissot(col(red%50))
```



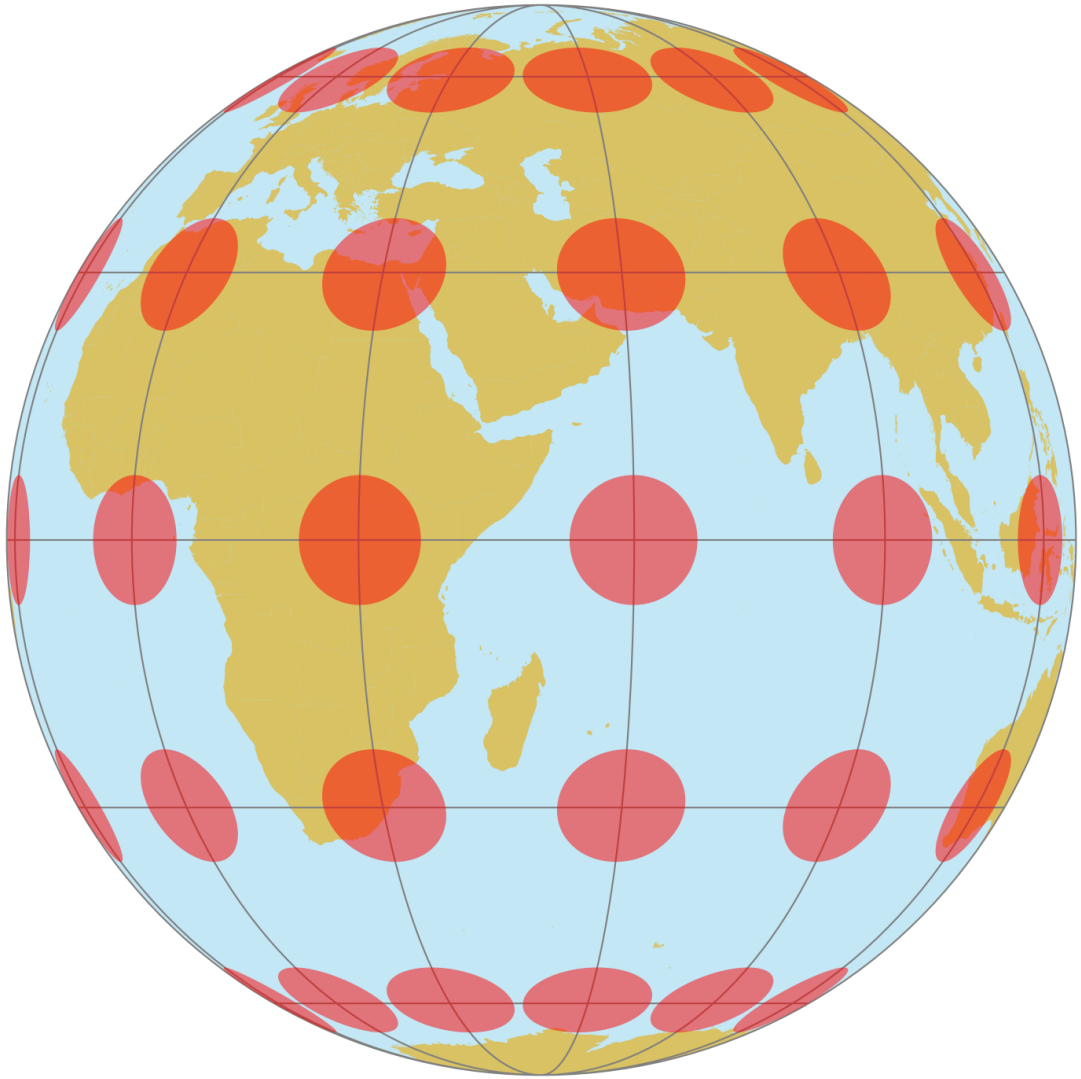
Orthographic projection (azimuthal)

```
geoplot (area world ECONOMY, color(viridis) lc(gray) lw(.1)), tight ///
    background(water) grid legend(position(sw)) ///
    margin(l=20) project(orthographic 1 50)
```



. . . with Tissot's indicatrices

```
geoplot (area world, col(sand)), tight ///
    background(water) grid margin(l=20) project(orthographic 1 50) ///
    tissot(col(red%50))
```



Data on Mexico from www.gits.igg.unam.mx/idea/descarga:

```
. geoframe create Estatal "Shapefile - Censo 2010 (Estatal).zip"
(translating Shapefile - Censo 2010 (Estatal).zip/inegi_refcenesta_2010.shp)
(importing shp file) (5 vars, 659,531 obs)
(importing dbf file) (190 vars, 32 obs)
(creating frame Estatal)
(creating frame Estatal_shp)
            Frame name: Estatal [make current]
            Frame type: attribute
          Feature type: <none>
         Number of obs: 32
               Unit ID: _ID
           Coordinates: _CX _CY
    Linked shape frame: Estatal_shp
. frame Estatal: geoframe simplify
(simplification threshold = .0000721)
(simplifying 312 shape items)
(0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(refinement threshold = .1827136)
(refining 85 shape items)
(0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(dropped 644,157 observations in frame Estatal_shp)
(added 196 observations in frame Estatal_shp)
```

Illustration of `inset()` option (can be repeated):

```
geoplot (area Estatal i._ID), nolegend ///
    inset(area world, lw(.1) color(sand) || area world if _ID==110, color(stc2) || ///
        , nobox size(40) pos(ne) title(Mexico is here) project(orthographic 1 -70) ///
           background(water lc(gray) limits(-180 180 -90 90)))
```

More data on Mexico from www.gits.igg.unam.mx/idea/descarga:

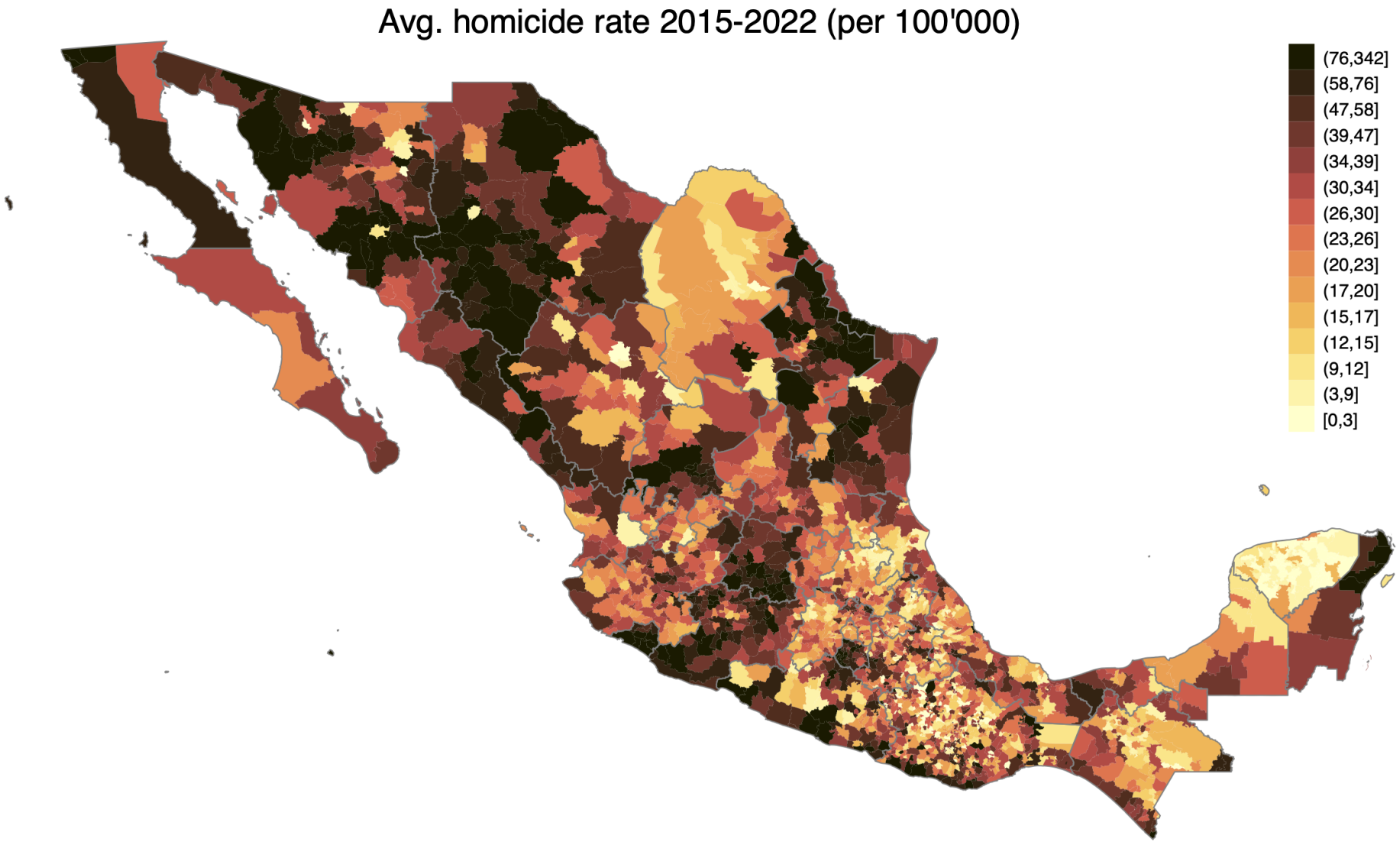
```
. geoframe create Municipal "Shapefile - Censo 2010 (Municipal).zip"
(translating Shapefile - Censo 2010 (Municipal).zip/inegi_refcenmuni_2010.shp)
(importing shp file) (5 vars, 3,283,138 obs)
(importing dbf file) (192 vars, 2,456 obs)
(creating frame Municipal)
(creating frame Municipal_shp)
            Frame name: Municipal [make current]
            Frame type: attribute
          Feature type: <none>
         Number of obs: 2,456
               Unit ID: _ID
           Coordinates: _CX _CY
    Linked shape frame: Municipal_shp
. frame Municipal: geoframe simplify
(simplification threshold = .0000721)
(simplifying 2862 shape items)
(0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(refinement threshold = .1827136)
(refining 2567 shape items)
(0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(dropped 3178096 observations in frame Municipal_shp)
(added 341 observations in frame Municipal_shp)
```

Add homicide data obtained from www.gob.mx:

```
. use Homicides, clear // (number of homicides and femicides in 2015-2022)
. frame Municipal {
.     destring cve_umun, replace
cve_umun: all characters numeric; replaced as int
.     geoframe copy default Homicides, id(cve_umun cvemunicipio)
(all units in frame Municipal matched)
(1 variable copied from frame default)
.     generate double hrate = Homicides/8 / (p_total/100000)
.     format %9.0f hrate
. }
```

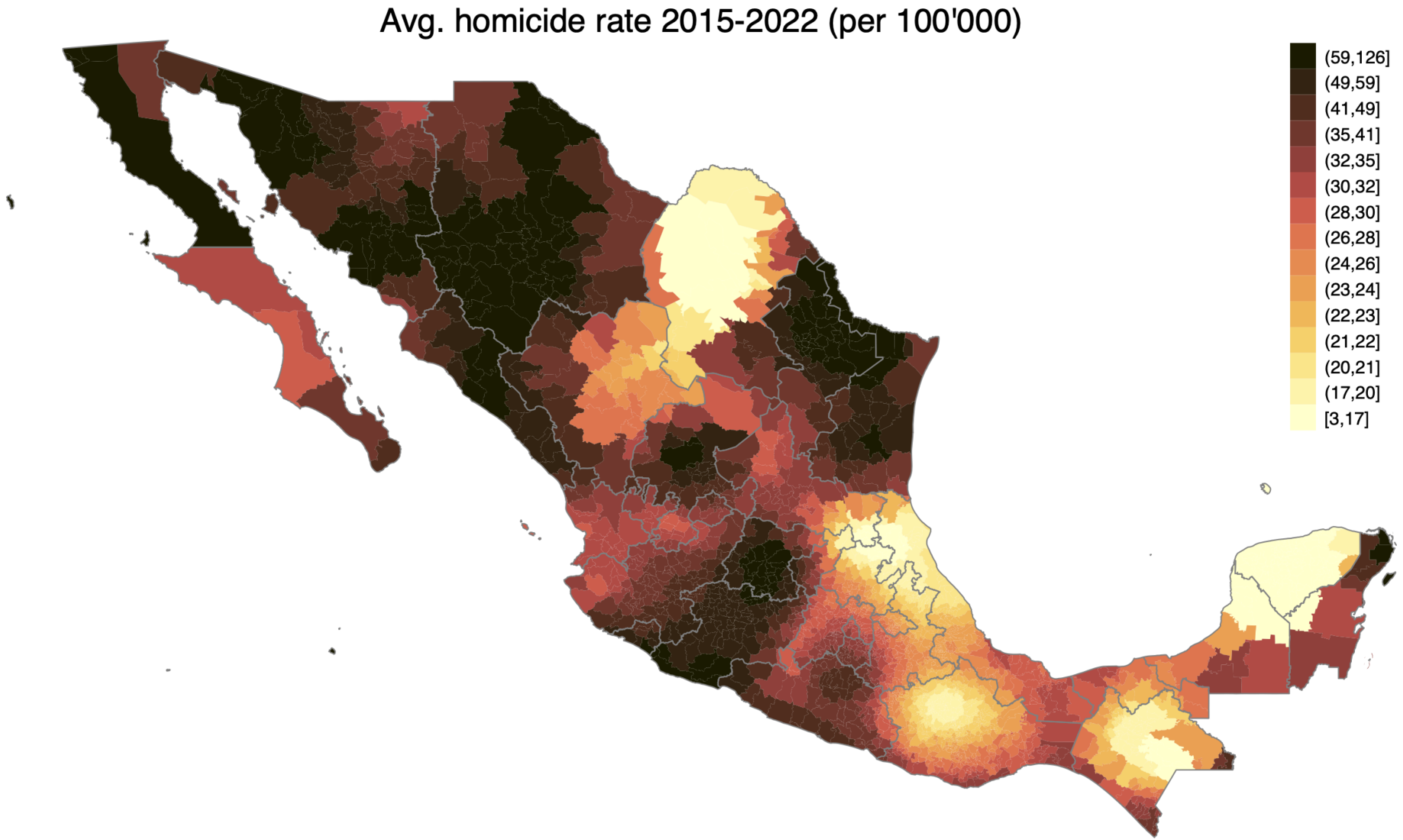
Homicide rate by municipality:

```
geoplot ///
    (area Municipal hrate, levels(15, quantile) color(scico lajolla)) ///
    (area Estatal), subtitle("Avg. homicide rate 2015-2022 (per 100'000)")
```



Apply smoothing:

```
frame Municipal: geoframe spsmooth hrate, generate(shrate)
geoplot ///
    (area Municipal shrate, levels(15, quantile) lab(, format(%9.0f)) color(scico lajolla)) ///
    (area Estatal), subtitle("Avg. homicide rate 2015-2022 (per 100'000)")
```

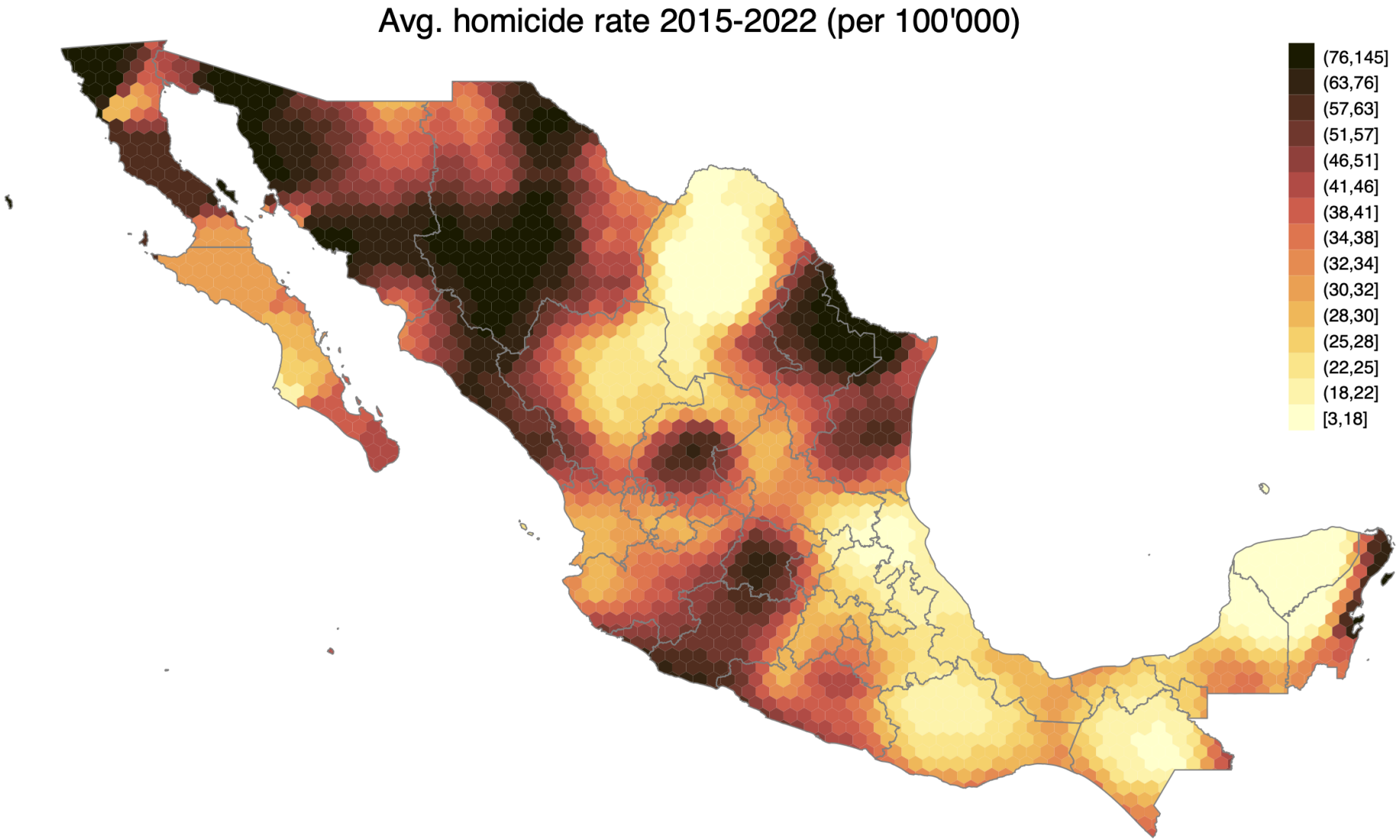


Generate raster:

```
frame Estatal: geoframe raster R, n(100) hex
geoplot (area R i.ID, fcolor(*.5)) (area Estatal), nolegend
```

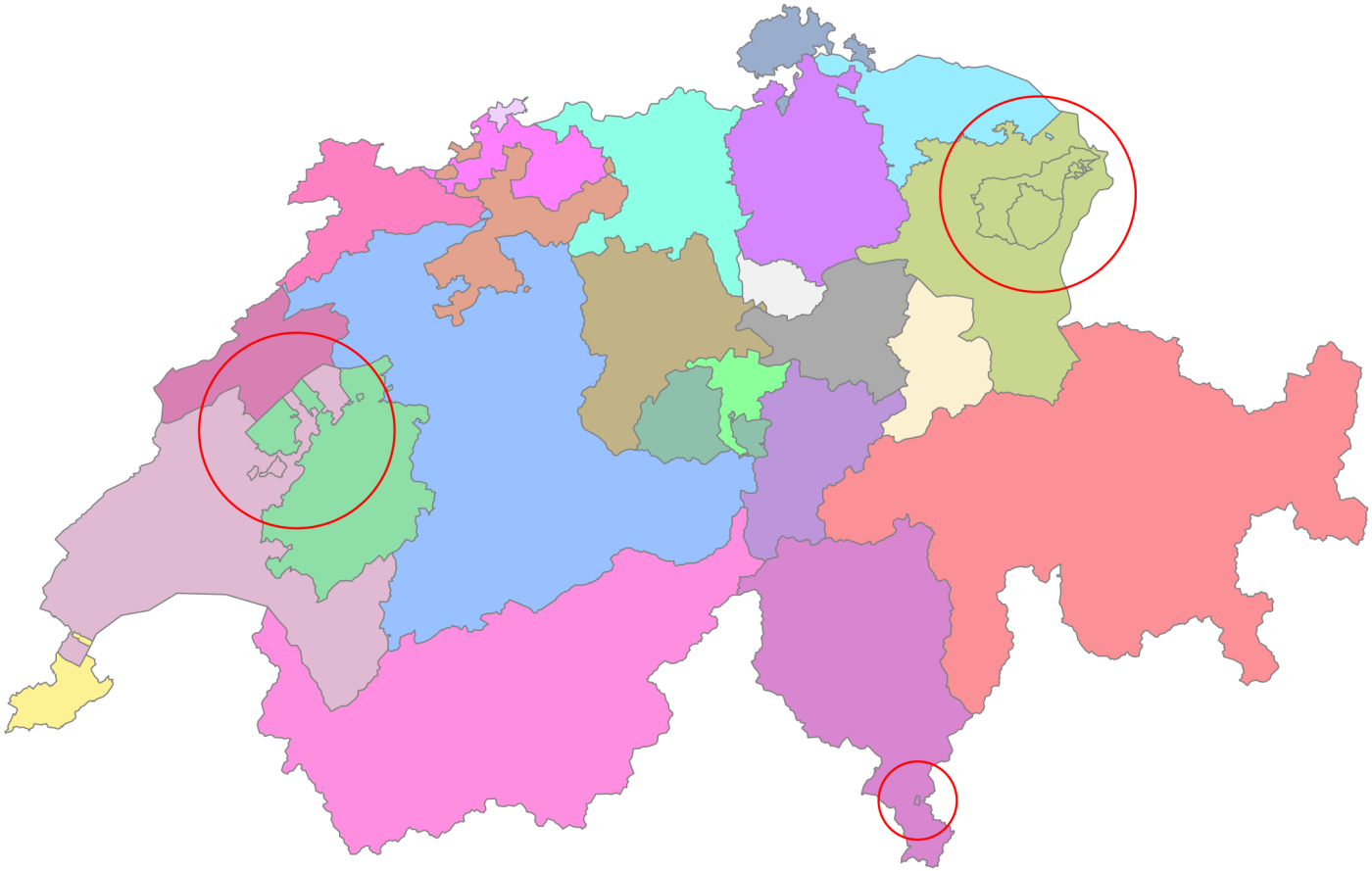
Smooth to raster:

```
frame Municipal: geoframe spsmooth hrate, at(R, fill)
geoplot ///
    (area R hrate, levels(15, quantile) lab(, format(%9.0f)) color(scico lajolla)) ///
    (area Estatal), subtitle("Avg. homicide rate 2015-2022 (per 100'000)")
```



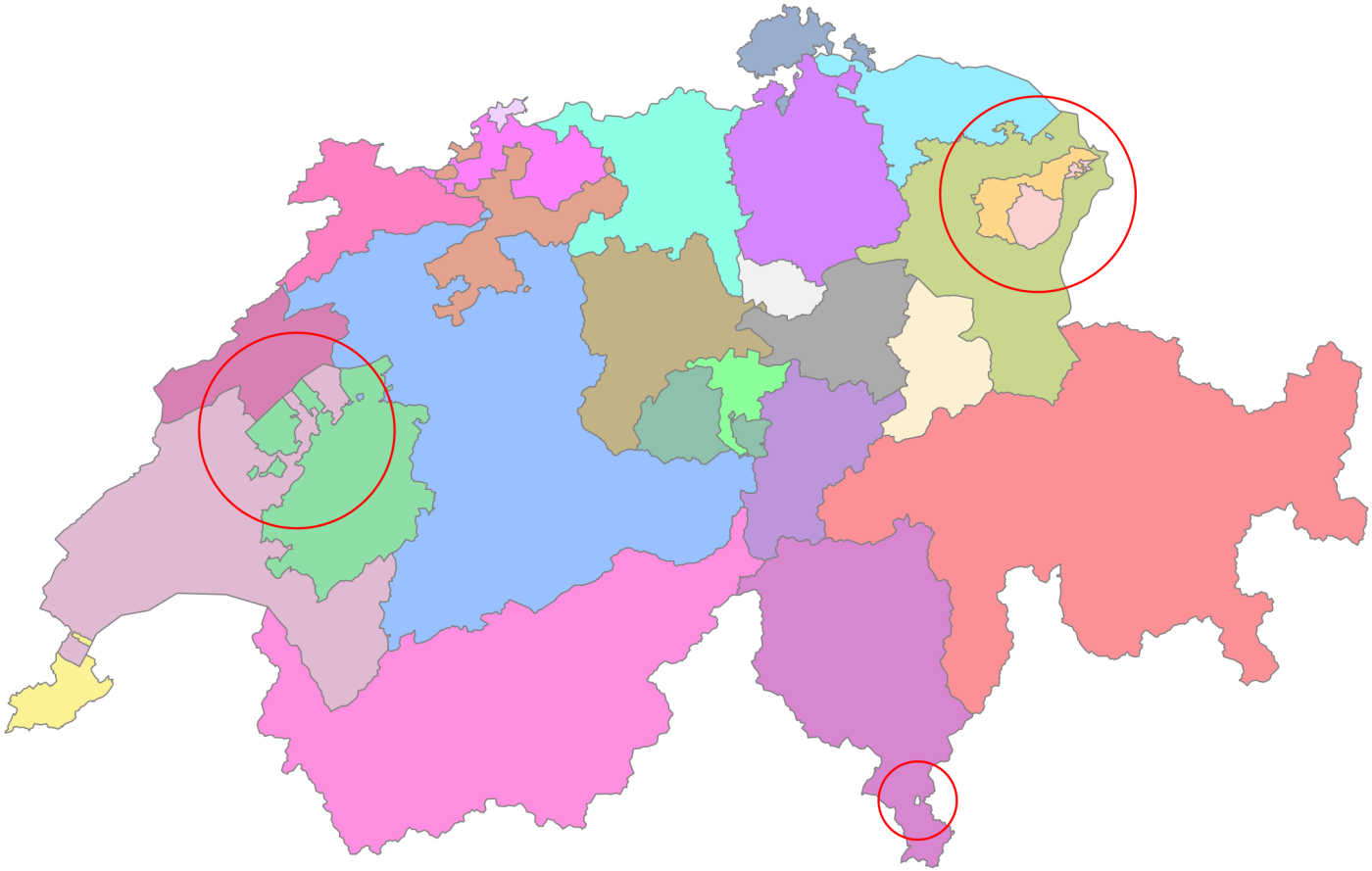
Swiss cantons: Some shapes are hidden

```
. geoframe create cantons, replace nodescribe
(creating frame cantons from cantons.dta)
(creating frame cantons_shp from cantons_shp.dta)
. local circles 2749490 1247410 [25000] 2560000 1187000 [25000] ///
>     2718751 1092403 [10000]
. geoplot (area cantons i._ID, color(alphabet2*.5)) (line cantons) ///
>     (symboli `circles',  lcolor(red) lwidth(medthin)),  nolegend
```



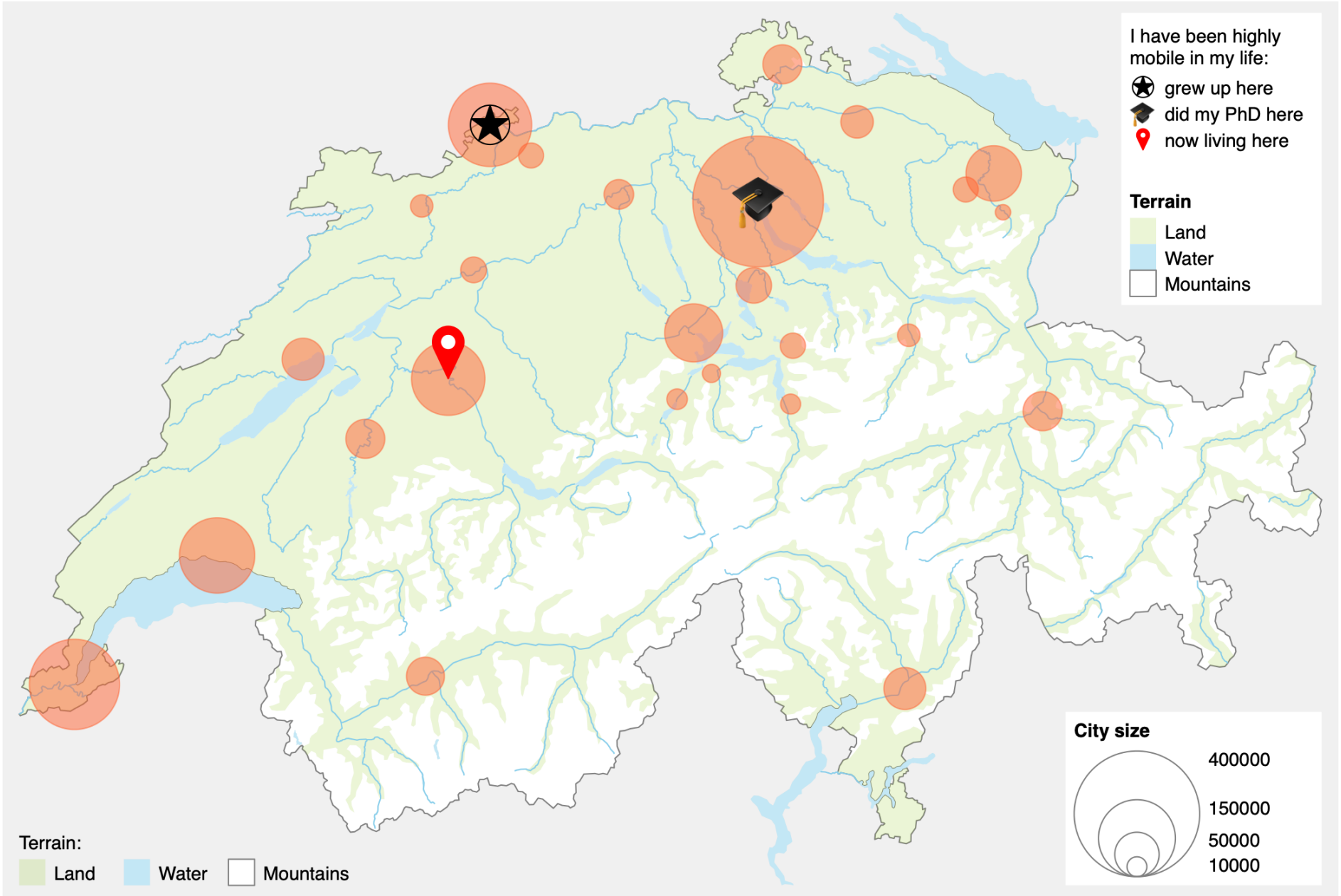
Identify enclaves/exclaves using `geoframe generate plevel`

```
. frame cantons: geoframe generate plevel
(pass 1/2: 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(pass 2/2: 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%)
(19 nested items found)
(variable _PLEVEL added to frame cantons_shp)
. geoplot (area cantons i._ID, color(alphabet2*.5)) (line cantons) ///
>     (symboli `circles',  lcolor(red) lwidth(medthin)),  nolegend
```

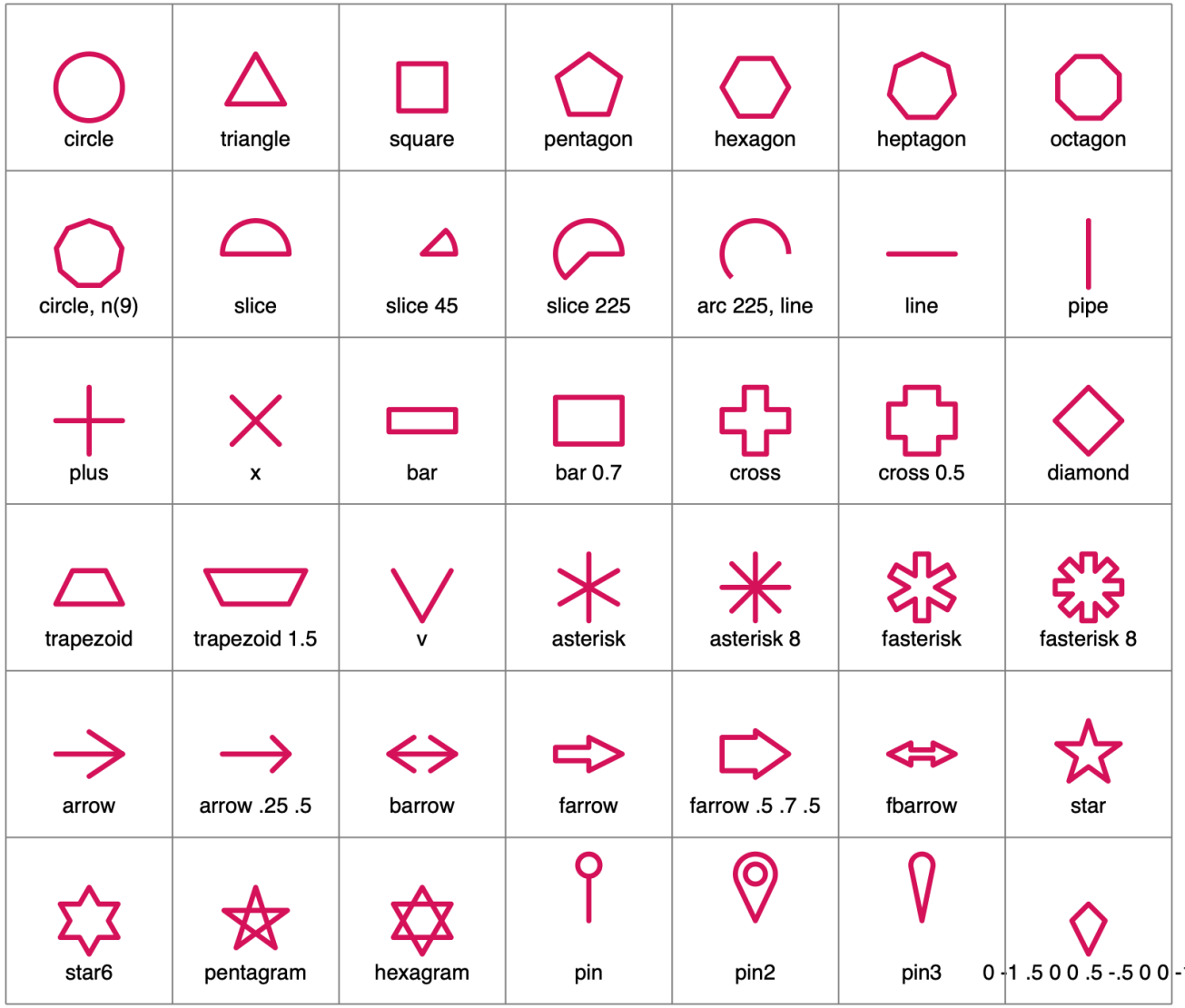


- Plot type `symbol` offers a variety of custom symbols; you can also define your own symbols and you can use unicode characters.
- Option `slegend()` can be used to illustrate sizes.
- Option `glegend()` provides a powerful and flexible alternative to the default legend option.

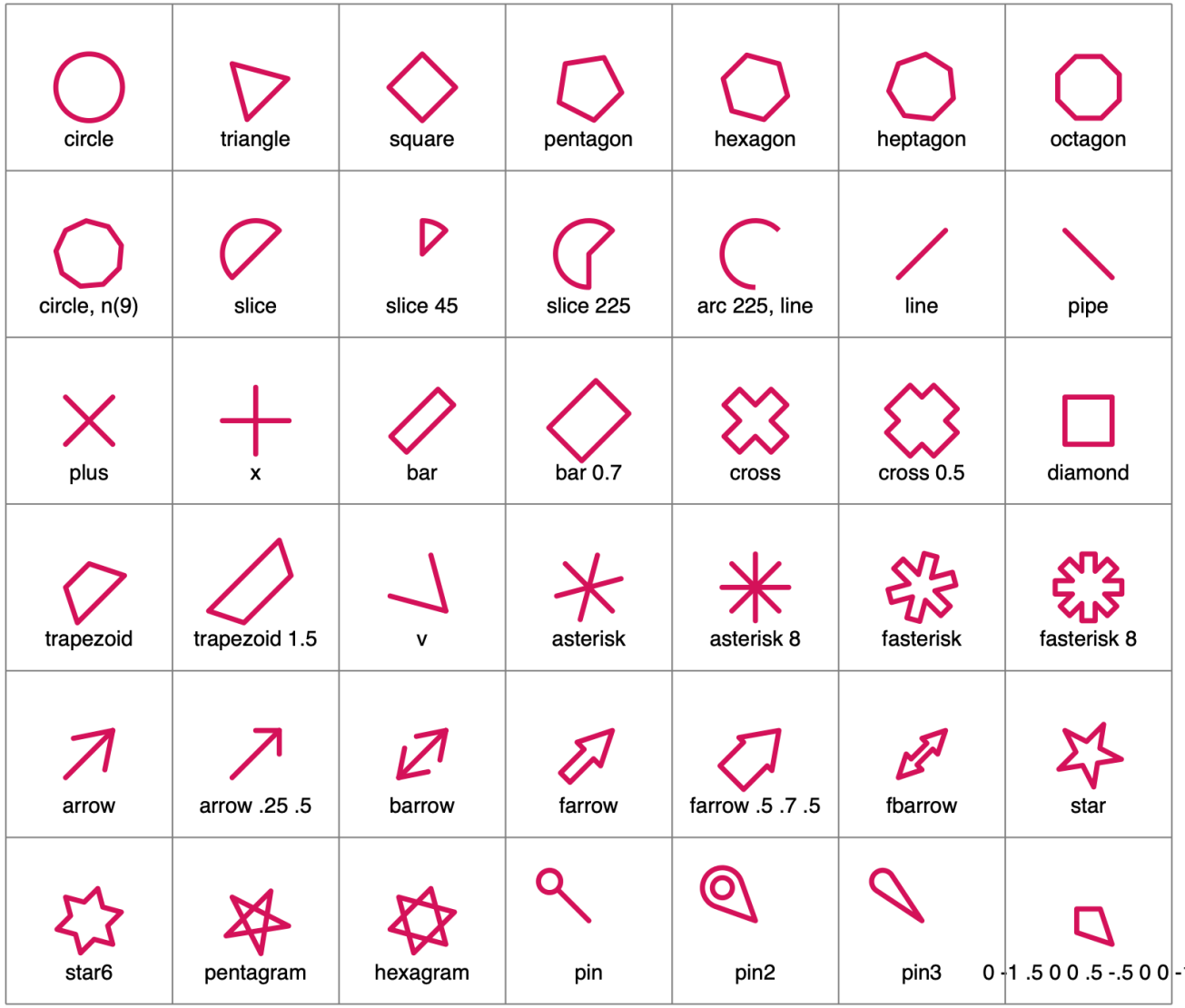
```
geoplot ///
    (area CH, if(_PLEVEL==0) fcolor(white)) ///
    (area CHvf, color(YellowGreen%20)) ///
    (area lakes) ///
    (line rivers) ///
    (symbol capitals [iw=bbtot], size(*5) color(stc6%50)) ///
    (symbol capitals (circle) if name=="Basel", size(*1.5) lcolor(black)) ///
    (symbol capitals (star)   if name=="Basel", size(*1.5) color(black)) ///
    (symbol capitals ("`=uchar(127891)'") if name=="Zürich", size(*2)) ///
    (symbol capitals (pin2) if name=="Bern", size(*2) color(red)) ///
    , bgcolor(gs15) tight ///
    slegend(1e4 5e4 15e4 4e5, overlay heading("{bf:City size}") ///
        position(se) box(color(white))) ///
    glegend(layout(- "I have been highly" "mobile in my life:" ///
        6&7 "grew up here" 8 "did my PhD here" 9 "now living here" ///
        . - "{bf:Terrain}" 2 "Land" 3 "Water" 1 "Mountains") ///
        lineskip(2.5) textwidth(17 10) box(color(white))) ///
    glegend(layout(- "Terrain:"  2 "Land" | 3 "Water" | 1 "Mountains") ///
        pos(sw) bottom textwidth(5))
```


I have been highly mobile in my life:
grew up here
did my PhD here
now living here
Terrain
Land
Water
Mountains
City size
400000
150000
50000
10000
Terrain:
Land
Water
Mountains

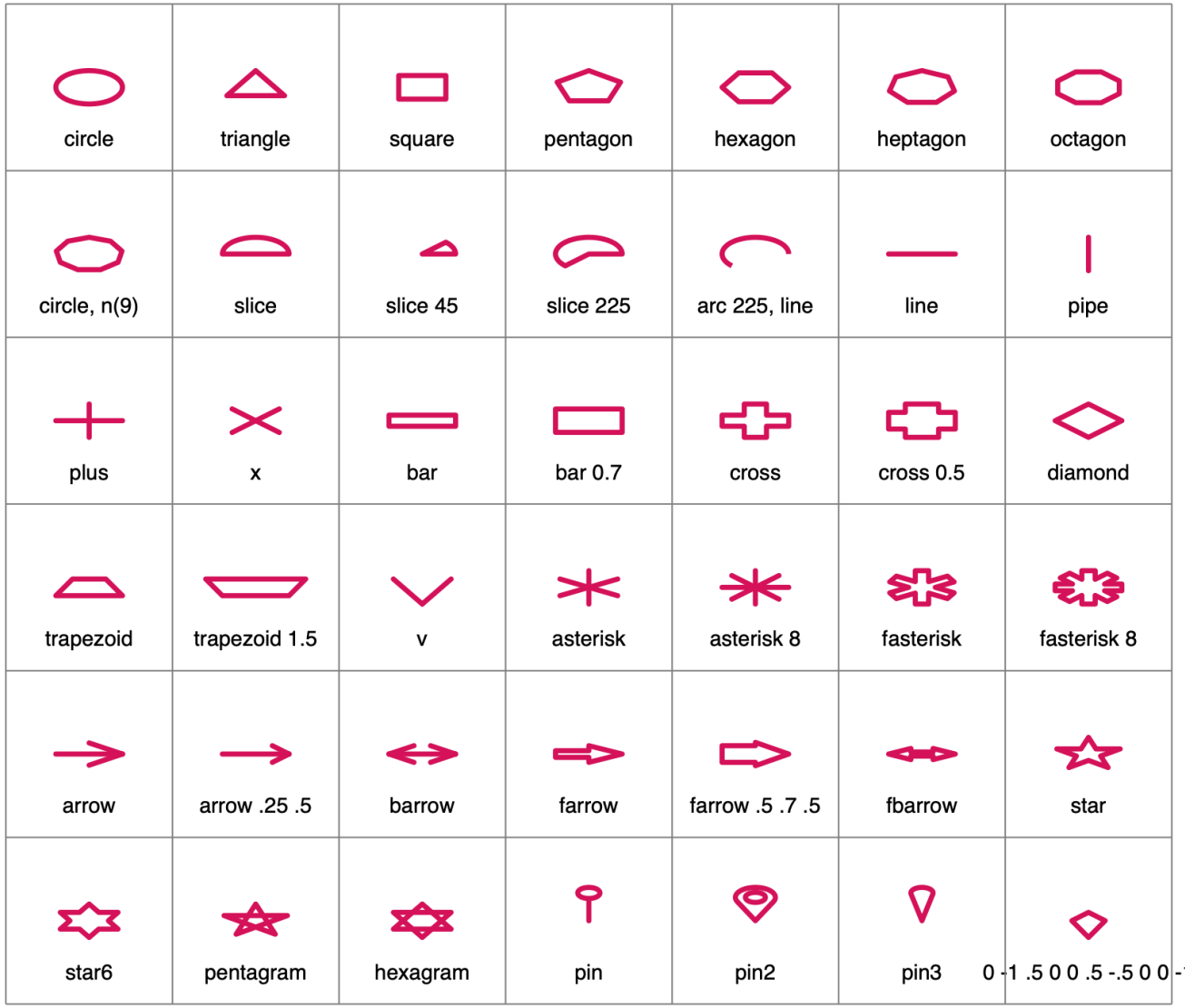
Symbols . . .


circle
triangle
square
pentagon
hexagon
heptagon
octagon
circle, n(9)
slice
slice 45
slice 225
arc 225, line
line
pipe
plus
x
bar
bar 0.7
cross
cross 0.5
diamond
trapezoid
trapezoid 1.5
v
asterisk
asterisk 8
fasterisk
fasterisk 8
arrow
arrow .25 .5
barrow
farrow
farrow .5 .7 .5
fbarrow
star
star6
pentagram
hexagram
pin
pin2
pin3
0 -1 .5 0 0 .5 -.5 0 0 -1

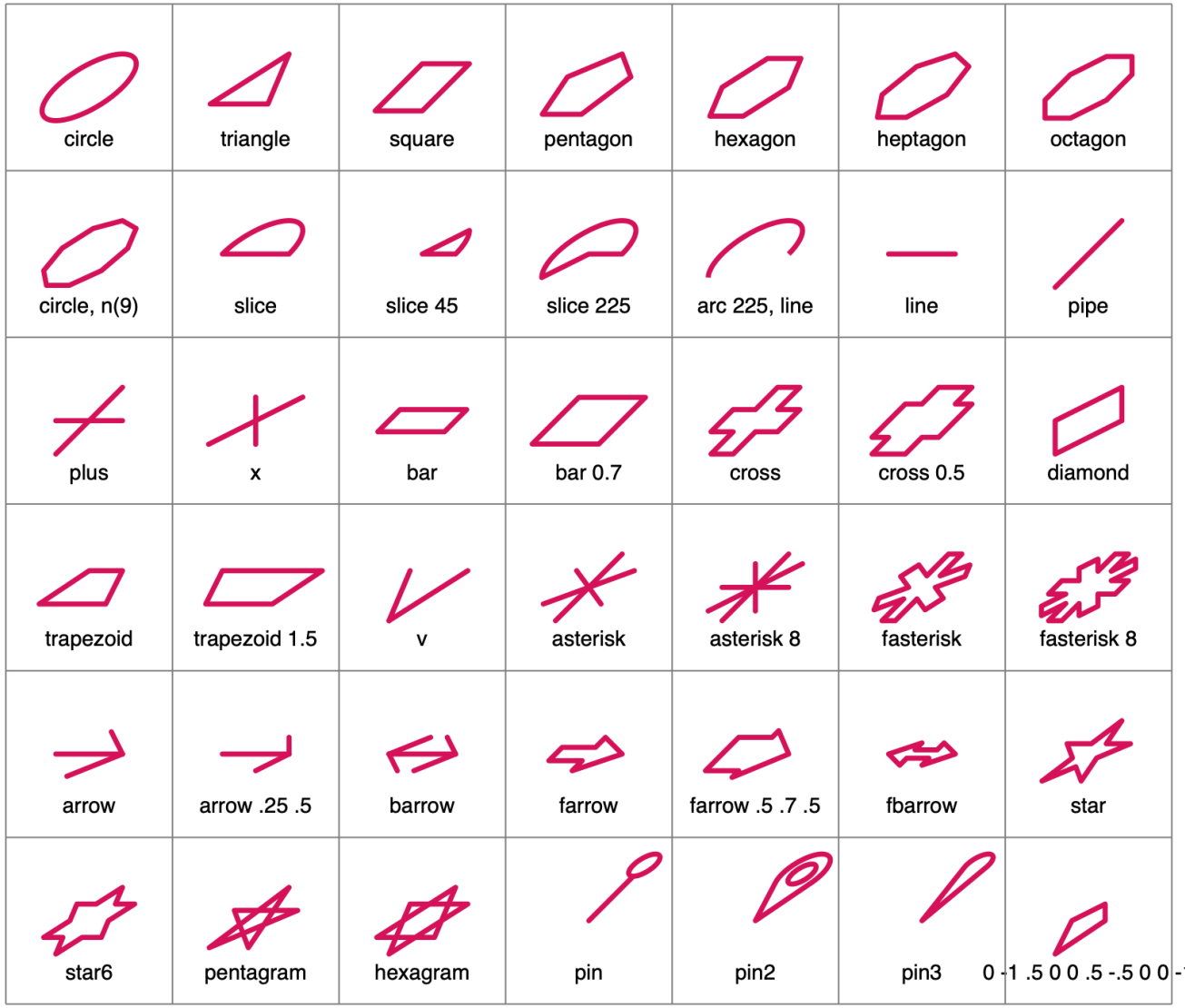
. . . with option `angle(45)`



... with option `ratio(0.5)`



... with option `slant(1)`



Conclusions

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

```
. ssc install geoplot, replace
. ssc install palettes, replace
. ssc install colrspace, replace
. ssc install moremata, replace
```

(Alternatively, get the latest `geoplot` update from github.com/benjann/geoplot.)

- Thanks to Asjad Naqvi for extensive testing and many suggestions.
- More examples:
  - README at github.com/benjann/geoplot.
  - Post by Asjad Naqvi at medium.com/the-stata-guide.
  - Some basic tutorials at doi.org/10.48350/188248.

Time for coffee!

```
. geoplot (area regions) ///
>    (symbol cities (star) if pop>5e5, color(hotpink) size(*2)) ///
>    (symbol cities ("`=uchar(9749)'")) ///
>    , glegend(layers(3 "coffee" 2&3 "great coffee") symsize(8) ///
>        symscale(1.4, common) box tsize(small) textwidth(17)) tight
```