Geocomputation with R

Reproducible Geo* workflows, from getting data to deploying map-based applications

Robin Lovelace, SatRdays Newcastle

2019-04-06. Source code: github.com/geocompr/geocompkg

What is this talk about?

This talk will introduce *Geocomputation with R*, a new book on R for geographic data. It will demonstrate how far R has evolved as an environment for geographic data analysis and visualisation, and provide a taster of what is in the book and, more importantly, what is possible when 'data science' and 'GIS' meet. With new packages such as **sf**, geographic data analysis has become more accessible. The talk will show how geographic analysis can bring local examples to life.

```
str(newcastle_geo_talk)
```

```
## List of 4
##  $ introduction: chr "introduction to geocomputation with R"
##  $ ecosystem   : chr "prominent spatial packages"
##  $ vizpkgs     : chr "visualisation packages"
##  $ fin         : chr "what next?"
```

intro

'Team geocompr'

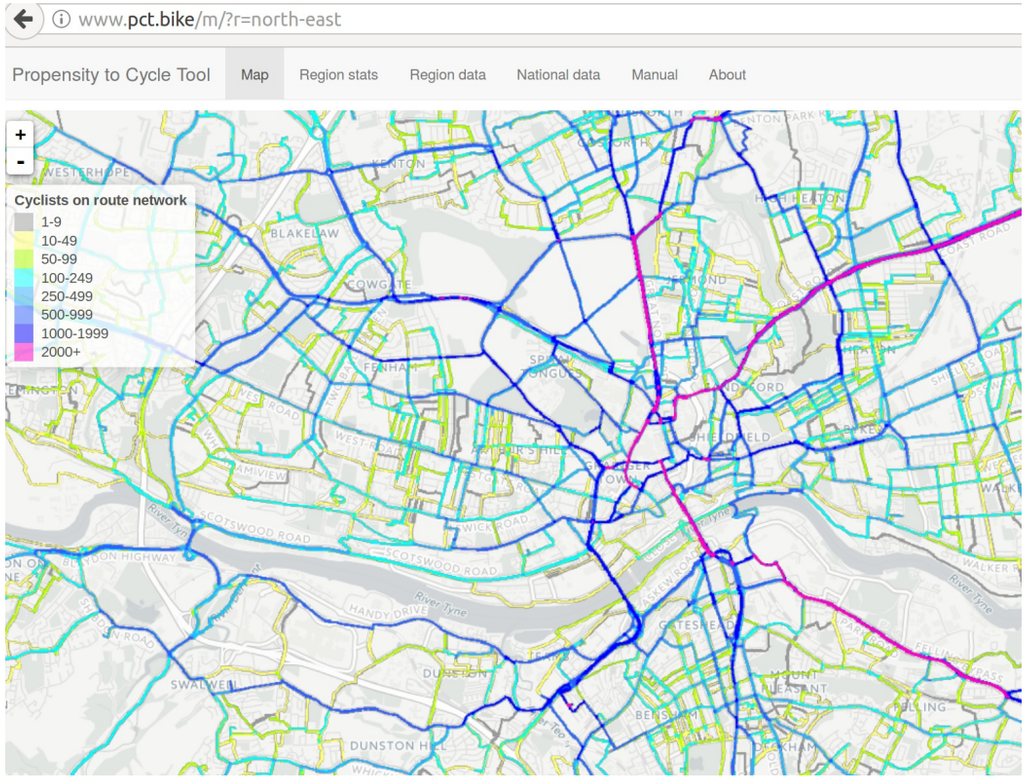
- Jakub Nowosad: developer of GeoPAT + more.
- Jannes Muenchow, creator of **RQGIS**.
- Robin Lovelace, creator of **stplanr**, co-author of Efficent R Programming.

```
library(leaflet)
popup = c("Robin", "Jakub", "Jannes'
m = leaflet() %>%
  addProviderTiles("NASAGIBS.ViirsEa
  addMarkers(lng = c(-3, 23, 11),
             lat = c(52, 53, 49),
             popup = popup)
```

```
m
```

Why use geographic data?

- Geographic data is everywhere
- underlies some of society's biggest issues
- Give generalised analyse local meaning



- Example: Propensity to Cycle Tool (PCT) in Newcastle: http://pct.bike/m/?r=north-east

What is Geocomputation?

> GeoComputation is about using the various different types of geodata and about developing relevant geo-tools within the overall context of a 'scientific' approach (Openshaw and Abrahart, 2000).

- But we do differ from early definitions in one important way:

> At the turn of the 21st Century it was unrealistic to expect readers to be able to reproduce code examples, due to barriers preventing access to the necessary hardware, software and data (Lovelace, Nowosad, and Meunchow, 2019)

Another definition

> about harnessing the power of modern computers to *do things* with geographic data.

What's in the geocompr box?

- Chapter 1: History + 'philosophy' = important

Foundations

- Starting from nothing
- Class definitions
- Spatial/attribute operations
- Projections
- Data IO

Extensions

- Advanced methods
- How to build your own functions

Applications

- A taster of what you can do

Context

- Software for 'data science' is evolving
- **tidyverse** packages **ggplot2** and **dplyr** have become immensely popular
- Shift in public-facing emphasis:
  - Away from implementation (computational efficiency)
  - Towards user friendliness (productivity)

There are different types of efficiency (source: Efficient R Programming)

> The second, broader definition of efficient computing is programmer productivity. This is the amount of useful work a person (not a computer) can do per unit time.


broom
dplyr
reshape2
%>%
Ceci n'est pas un pipe.
test that
purrr
www.rstudio.com
rmarkdown
RStudio
knitr
Shiny
packrat
lubridate
ggplot2
haven
stringr
tidyr
readr

ecosystem

A brief history of geographic packages in R

- See a video of Roger Bivand's talk on the subject (live demo of R 0.49, released 1997) + GitHub repo
- Brian Ripley, on R's core team, with strong interest in spatial stats
- Rs capabilities have evolved substantially since then!
- **rgdal**, released in 2003
- **sp**, released 2005
    - **sp**verse: **gstat**, **geosphere**, **adehabitat**
- **raster**, released 2010

Viz developments

"The core R engine was not designed specifically for the display and analysis of maps, and the limited interactive facilities it offers have drawbacks in this area" (Bivand, Pebesma, and Gomez-Rubio, 2013).

Five years later...

"An example showing R's flexibility and evolving geographic capabilities is **leaflet** , a package for making interactive maps that has been extended by the R community, as we'll see in Chapter 9" (Lovelace, Nowosad, and Muenchow, 2018).

Going way back...

R's predecesor was S, which was itself inspired by lisp (Chambers, 2016).

This is geographic analysis in S (Rowlingson and Diggle, 1993):

```
pts <- spoints(scan('cavities'))
uk()
pointmap(pts,add=T)
zoom()
uk(add=T)
pointmap(pts,add=T)
poly<-getpoly()
```

Still works today, 25 years later:

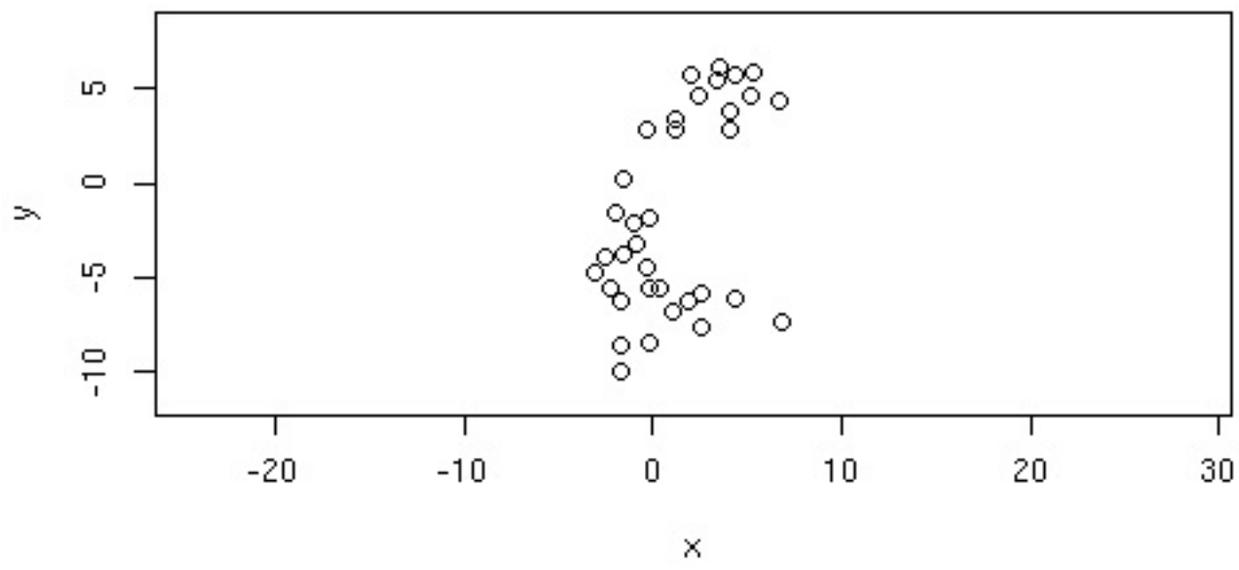
```
library(splancs)
```

```
## Loading required package: sp

##
## Spatial Point Pattern Analysis Code in S-Plus
##
##  Version 2 - Spatial and Space-Time analysis
```

```
#> Spatial Point Pattern Analysis Code in S-Plus
#> Version 2 - Spatial and Space-Time analysis
```

splancs

```
# install.packages("splancs"); library(splancs)
# example, interactive! (commented bits)
data(bodmin)
plot(bodmin$poly, asp=1, type="n")
pointmap(as.points(bodmin), add=TRUE)
```




```
# zoom()
# pointmap(as.points(bodmin), add=TRUE)
```

Observations

- R' is robust and future-proof
- It was hard
- But it still worked

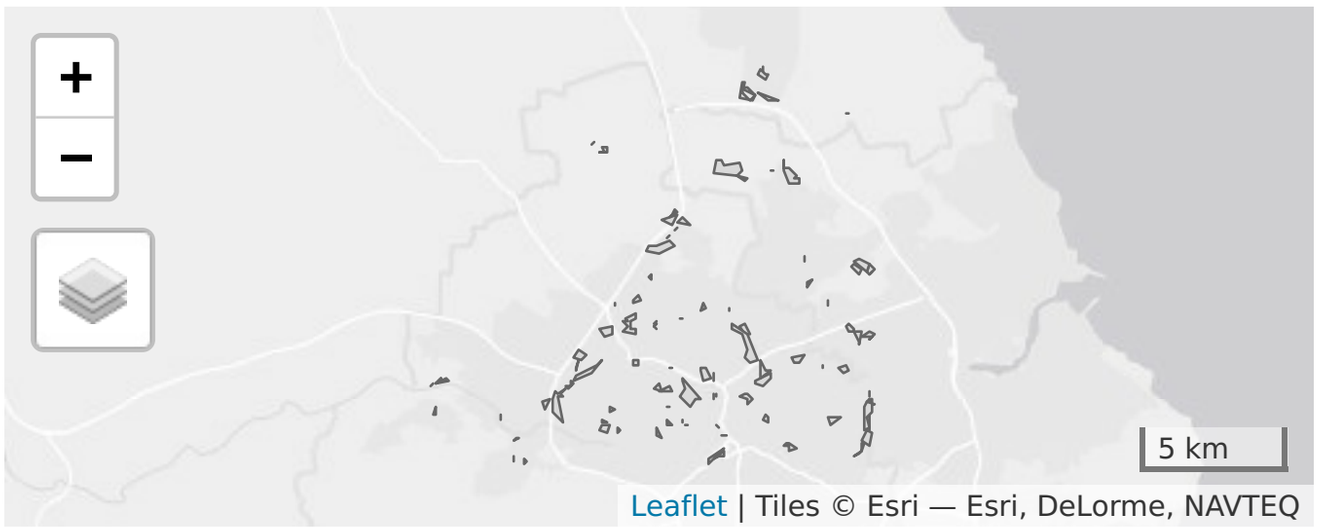
Getting data

- Many portals and packages for getting data - see Chapter 7

```
library(osmdata)
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. http://www.openstreetmap.org/copyr
```

```
newcastle = getbb("newcastle")
osm_data = opq(bbox = newcastle) %>%
  add_osm_feature(key = "leisure", value = "park") %>%
  osmdata_sf()
parks = osm_data$osm_polygons
```



Getting transport data

- pct package developed to make transport data more accessible
- Get transport zones for Newcastle

```
zones_all = pct::get_pct_zones("north-east")
zones = zones_all[c("geo_code", "car_driver", "foot", "bicycle")]
plot(zones)
```

```
## Warning in polypath(p_bind(L), border = border[i], lty = lty[i], lwd =
## lwd[i], : Path drawing not available for this device

## Warning in polypath(p_bind(L), border = border[i], lty = lty[i], lwd =
## lwd[i], : Path drawing not available for this device

## Warning in polypath(p_bind(L), border = border[i], lty = lty[i], lwd =
## lwd[i], : Path drawing not available for this device

## Warning in polypath(p_bind(L), border = border[i], lty = lty[i], lwd =
## lwd[i], : Path drawing not available for this device

## Warning in polypath(p_bind(L), border = border[i], lty = lty[i], lwd =
## lwd[i], : Path drawing not available for this device

## Warning in polypath(p_bind(L), border = border[i], lty = lty[i], lwd =
## lwd[i], : Path drawing not available for this device
```

sf

- **sf** is a recently developed package for spatial (vector) data
- Combines the functionality of three previous packages: **sp**, **rgeos** and **rgdal**
- Has many advantages, including:
  - Faster data I/O
  - More geometry types supported
  - Compatibility with the *tidyverse*

Reading and writing spatial data

- `write_sf()` writes data `st_write(zones, 'zones.gpkg')`.
- See supported formats with: `sf::st_drivers()`. Details: Chapter 6 of our book: geocompr.robinlovelace.net/read-write.html

sf class

```
library(spData) # pre-packaged data
class(zones)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
zones
```

```
## Simple feature collection with 300 features and 4 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -2.355518 ymin: 54.45115 xmax: -0.7884347 ymax: 55.07939
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 300 x 5
##    geo_name     all  foot bicycle                                    geometry
##    <chr>      <int> <int>   <int>                          <MULTIPOLYGON [°]>
##  1 Gateshead…  3428   229      58 (((-1.744109 54.98028, -1.728493 54.9749…
##  2 Gateshead…  4300   230      57 (((-1.7651 54.98013, -1.76156 54.97866, …
##  3 Gateshead…  2954   277      70 (((-1.530445 54.95821, -1.530117 54.9577…
##  4 Gateshead…  3406   321      30 (((-1.707878 54.9583, -1.711479 54.94908…
##  5 Gateshead…  3226   233      28 (((-1.701471 54.97029, -1.679887 54.9635…
##  6 Gateshead…  3525   397      95 (((-1.625103 54.94904, -1.619229 54.9454…
##  7 Gateshead…  3042   693      78 (((-1.609381 54.96109, -1.608685 54.9580…
##  8 Gateshead…  3419   392      36 (((-1.662861 54.95036, -1.661506 54.9455…
```

tidyverse

```r
library(tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────

## ✔ ggplot2 3.1.1       ✔ purrr   0.3.2
## ✔ tibble  2.1.1       ✔ dplyr   0.8.0.1
## ✔ tidyr   0.8.3       ✔ stringr 1.4.0
## ✔ readr   1.3.1       ✔ forcats 0.4.0

## ── Conflicts ────────────────────────────────────────────
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ✖ dplyr::tribble() masks tibble::tribble(), splancs::tribble()
```

- Mostly compatible, but there are some pitfalls

Chaining geographic operations

```
parks_buff = parks %>%
  st_transform(crs = 27700) %>%  # uk CRS
  st_buffer(50) %>%
  st_transform(crs = 4326)
near_parks = zones[parks_buff, ] # spatial subsetting
zones$near_parks = lengths(st_intersects(zones, parks_buff)) > 0
```

```
qtm(zones, "near_parks")
```

```
## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
```

Gotchas with geographic data

- Some objects have multiple geometries:

```
unique(st_geometry_type(zones))
```

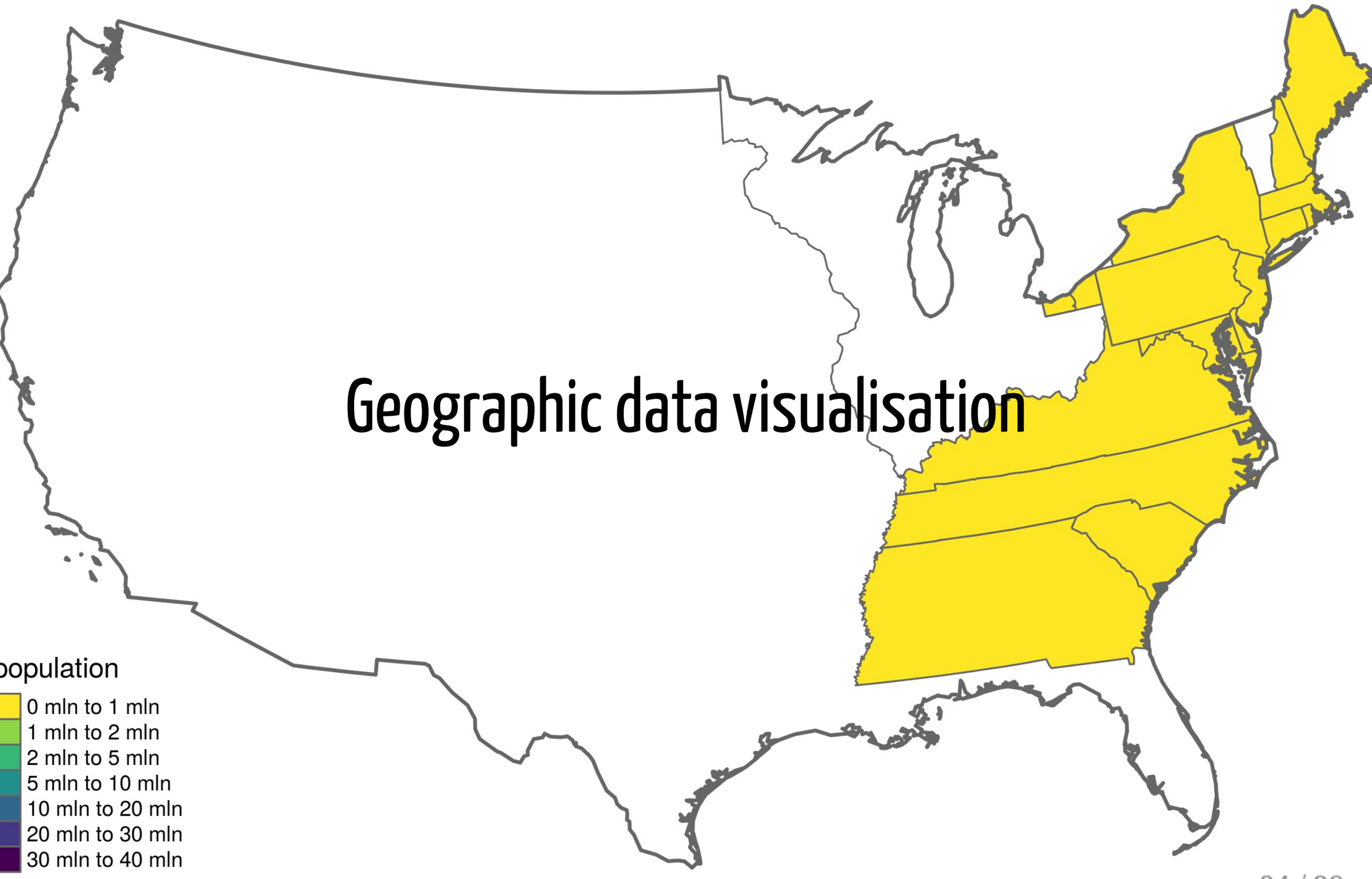
```
## [1] MULTIPOLYGON
## 18 Levels: GEOMETRY POINT LINESTRING POLYGON ... TRIANGLE
```

```
zones_POLYGON = st_cast(zones, "POLYGON")
unique(st_geometry_type(zones_POLYGON))
```

```
## [1] POLYGON
## 18 Levels: GEOMETRY POINT LINESTRING POLYGON ... TRIANGLE
```

- Coordinate references systems

Geographic data visualisation



sf graphics

```
library(sf)
# plot(zones[c("car_driver", "foot")])
```

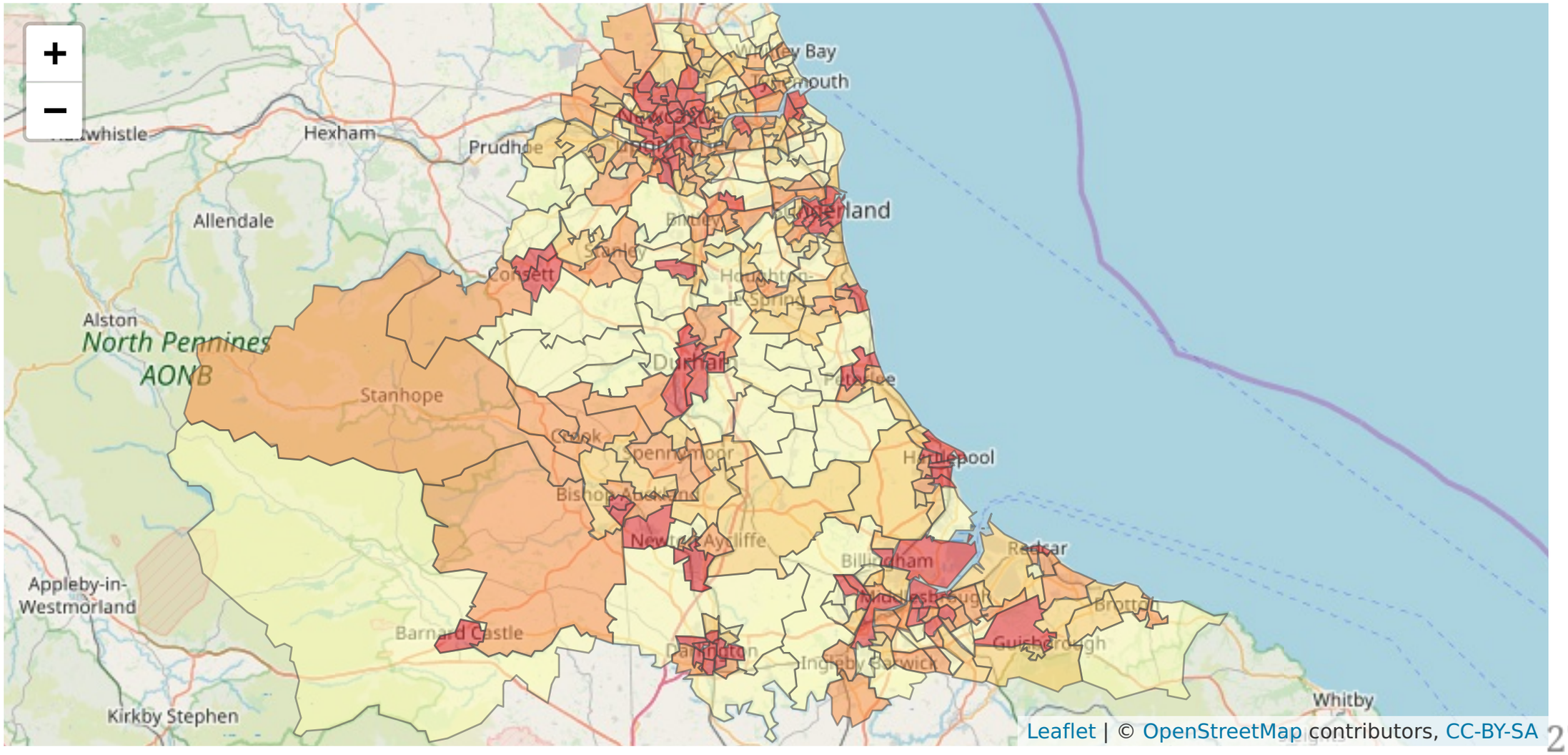
ggplot2

```
ggplot(zones) +
  geom_sf(aes(fill = bicycle)) +
  scale_fill_viridis_c()
```

```
## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
```

leaflet

```
library(leaflet)
leaflet(zones) %>%
        addTiles() %>%
        addPolygons(color = "#444444", weight = 1, fillOpacity = 0.5,
                    fillColor = ~colorQuantile("YlOrRd", bicycle)(bicycle),
                    popup = paste(round(zones$bicycle, 2)))
```



tmap

- A dedicated mapping R package

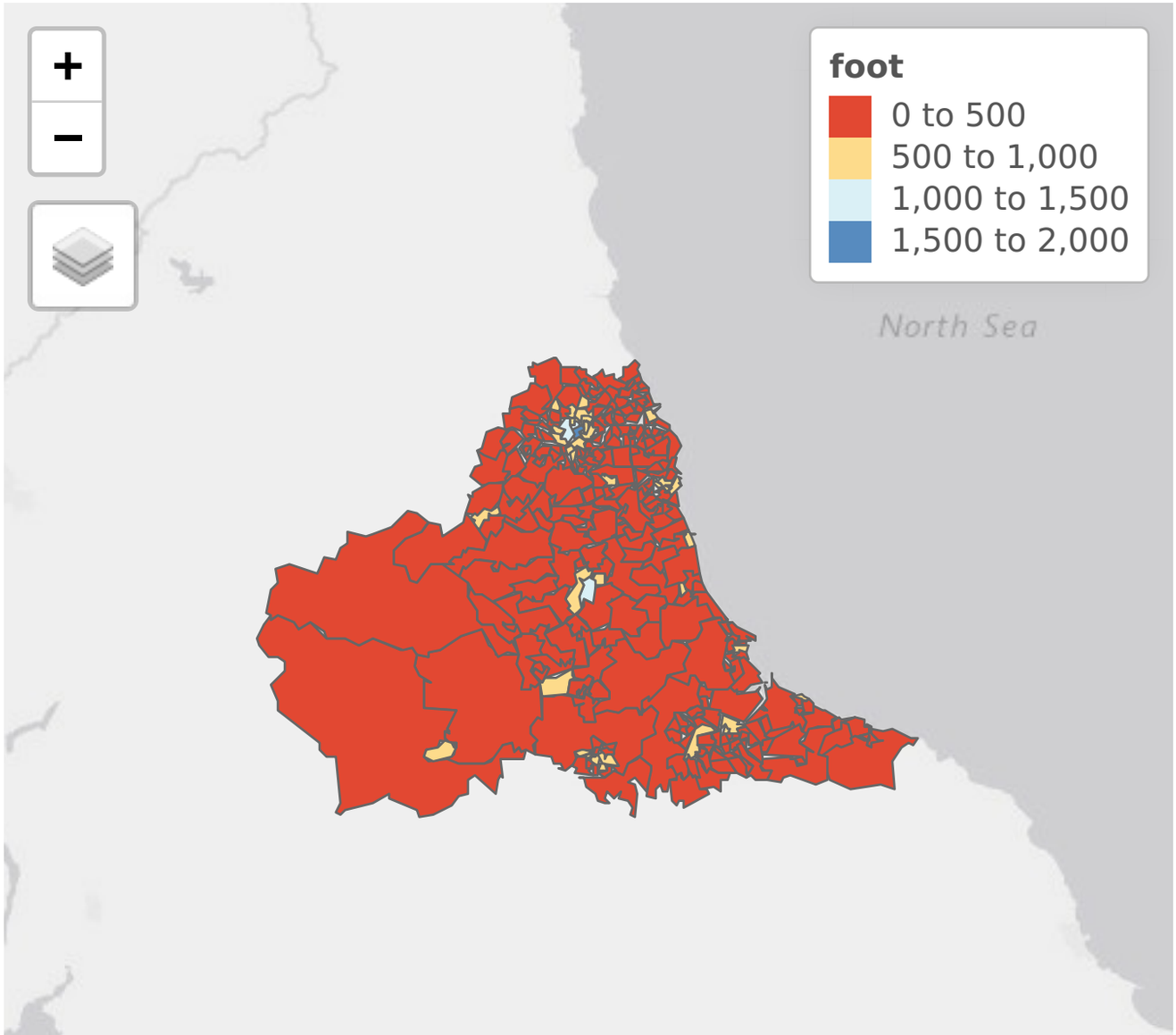
```
library(tmap)
tm_shape(zones) +
  tm_polygons("foot", palette = "RdYlBu")
```

```
## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device
## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device

## Warning in grid.Call.graphics(C_path, x$x, x$y, index, switch(x$rule,
## winding = 1L, : Path drawing not available for this device
```

tmap II

- A dedicated mapping R package

```
tmap_mode("view")
tm_shape(zones) +
  tm_polygons("foot", palette = "RdYlBu")
```



Mapdeck

- See example at geocompr.robinlovelace.net/adv-map.html

```
library(mapdeck)
set_token(Sys.getenv("MAPBOX"))
mapdeck(data = zones) %>%
  mapdeck::add_sf() %>%
  add_polygon(elevation = zones$foot^2)
```

Shiny

Here's one I made earlier: www.pct.bike

Geoplumber

GitHub package developed by Layik Hama - see github.com/ATFutures

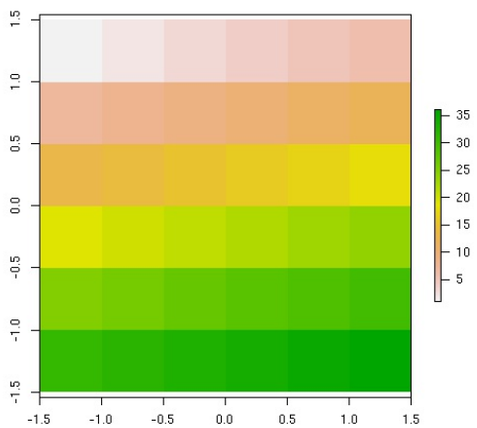
What next?

Advances in raster data with R

Raster data in R is evolving:

- **raster** package dominates, designed for `Spatial*` objects, e.g. with `as(my_vector, "Spatial")`
- Efforts to bring raster data closer to the **tidyverse**/**sf** approaches, such as tabularaster, sfraster + fasterize
- stars focusses on multidimensional space-time (raster/vector) is evolving.
- **terra** - first release expected 2019
- See https://github.com/edzer/sdsr

```
raster::plot(spData::elev)
```



Spatial networks

- Lots of packages out there, including `stplanr`, `dodgr` and `tidygraph`
- Source: https://github.com/spnethack/spnethack

Geographic 'leverage points' where R can help

A few examples:

- Visualising data: e.g geoplumber
- To the public - web viz pkgs help with that!
- Local issues. E.g. where's the most dangerous/unpopular road in Newcastle?
- Community cohesion

That is a balance R is ideally set-up to strike

Summary: do things with geographic data in R

Base R + **tidyverse** now (mostly) works with spatial data.

This is thanks to **sf**, a recent package (first release in 2016) that implements the open standard data model *simple features*. Get **sf** with:

```
install.packages("sf")
```

Raster data is also supported, in the more mature package **raster**:

```
install.packages("raster")
```

For datasets...:

```
install.packages("spData")
install.packages("rnaturalearth")
```

For more on this see the book: github.com/Robinlovelace/geocompr

- 2 day workshop: lida.leeds.ac.uk/event

Thanks, links, happy R day travels , + !

- Reproducible slides: [githttps://github.com/geocompr/geocompkgttps://github.com/Robgeocompr/geocompkg-
Geocomputation (feedback welcome): geocompr.robinlovelace.net

Slides created via the R package **xaringan**.

CDRC TRAINING / APR 25 @ 9:30 AM - APR 26 @ 4:30 PM

Geocomputation and Data Analysis with R

The aim of Geocomputation and Data Analysis with R is to get you up-to-speed with high performance geographic processing, analysis, visualisation and modelling capabilities from the command-line. The course will be delivered in R, a statistical programming language popular in academia, industry and, increasingly, the public sector. It will teach a range of techniques using recent developments in the package sf and the 'metapackage' tidyverse, based on the open source book Geocomputation with R (Lovelace, Nowosad, and Meunchow 2019).