Maps

EDC101: Week 8

Dr. Calum Webb
Sheffield Methods Institute, the University of Sheffield
c.j.webb@sheffield.ac.uk

Sign in

Learning outcomes

What will I learn?

By the end of this week you will know:

- How to create visualisations of spatial data (maps) in R.
- How to interpret these maps and tell a story with them to engage your audience.
- How to join relational data in order to combine spatial and other forms of data files.
- How to critically evaluate the use of maps in data visualisation: when maps are and are not appropriate, and what ideas about society they perpetuate.

Week 8: Maps — Part I

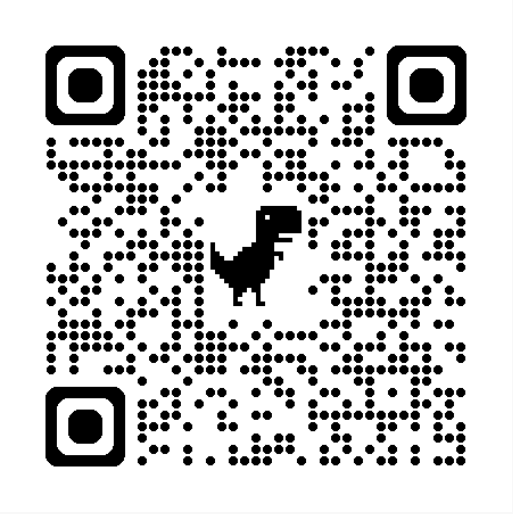
Why do we visualise data in the form of maps?

How have the most popular areas for buying houses in Sheffield changed over time?

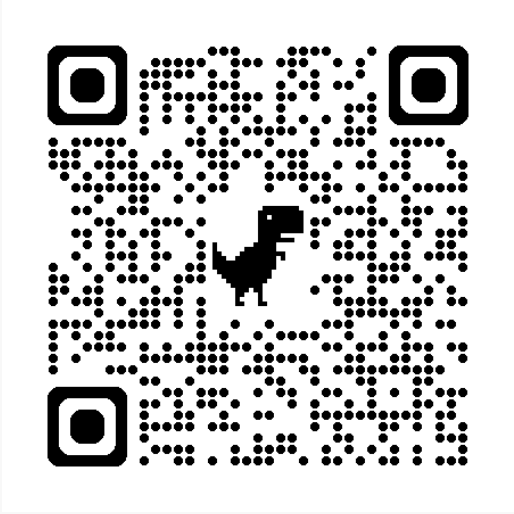
The two links below will take you to interactive maps of Sheffield where each small area has been colour coded according to the quantity of **houses sold during the year**. House sales have been grouped into 20 quantiles, meaning that an area being in quantile 20 means **it's in the 5% highest sales of homes in the city** and an area being in quantile 1 means **it's in the 5% lowest sales of homes in the city**.

Has the area you live in gotten more or less popular for house buyers over that 20 year time span?

House Sales in 1998: bit.ly/sheff-1998



House Sales in 2018: bit.ly/sheff-2018

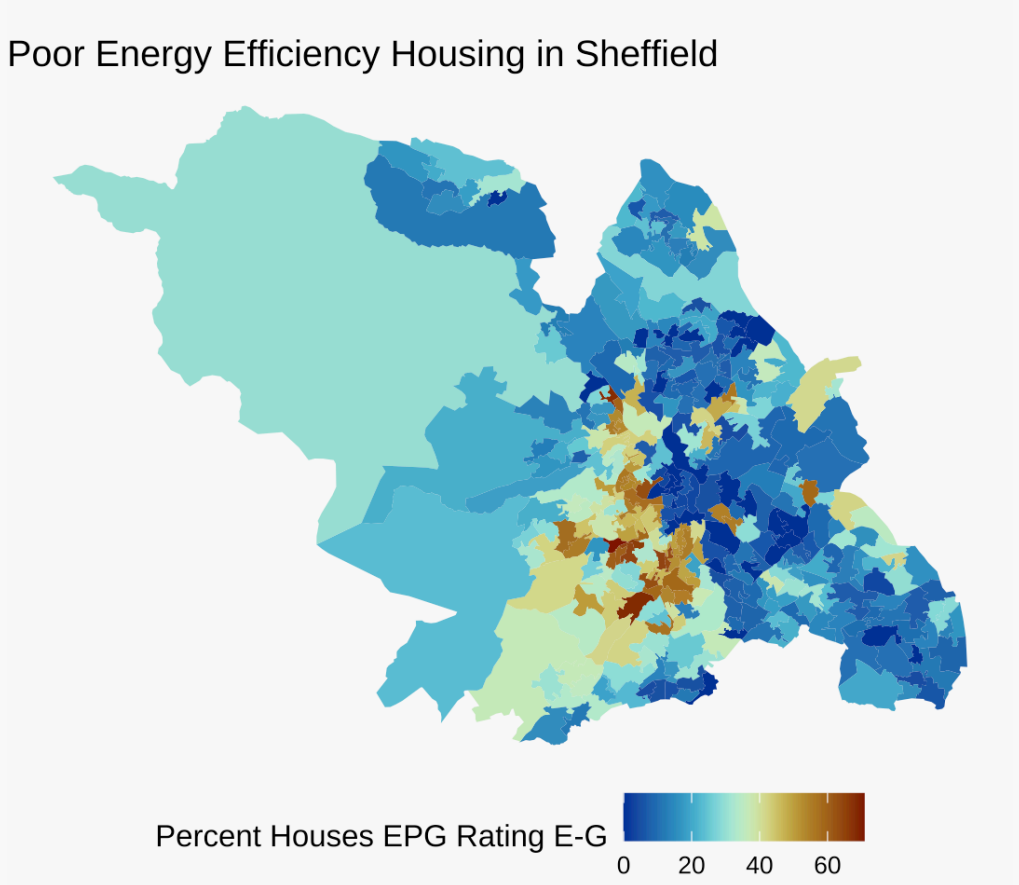


Why might we want to use maps in our data visualisation?

Argument 1: Because it's interesting.

> If we know this data is referencing specific places, why do we throw away that information when we turn it into a scatterplot, etc.

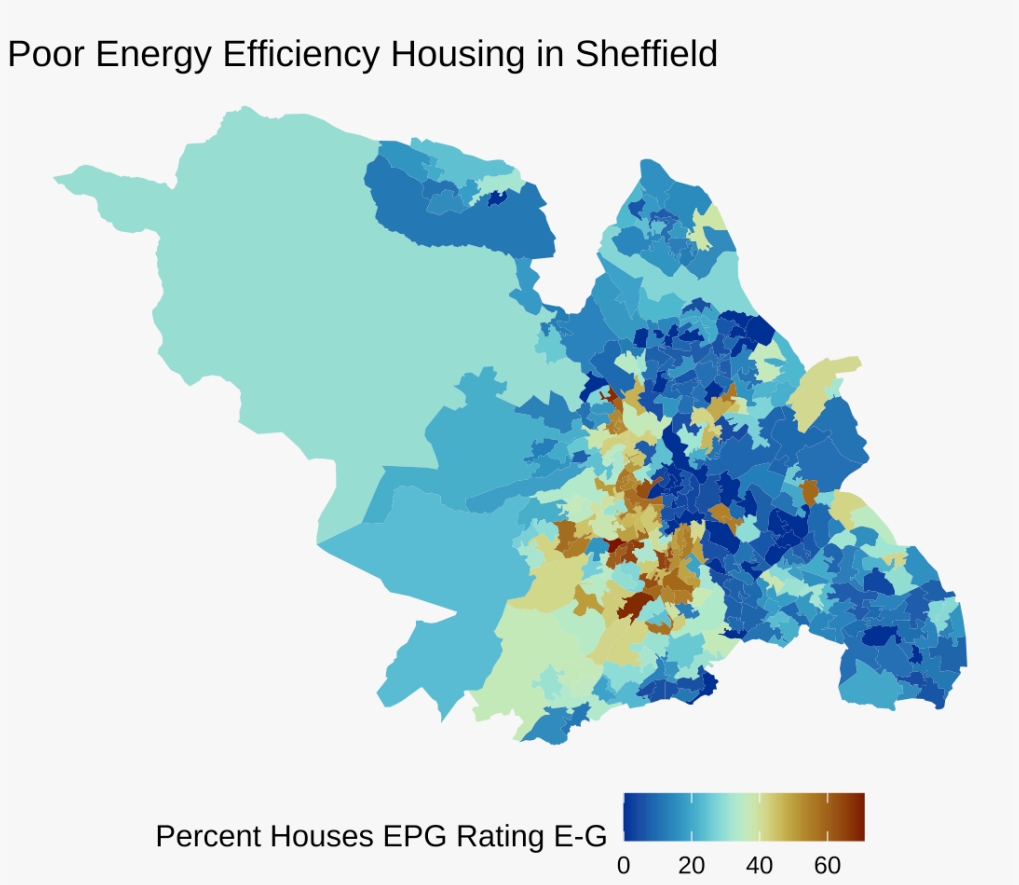
- Often an engaging way to present data — few people are interested in general patterns, everyone's interested in where they live.
- Can show interesting things in its own right — concentration or dispersion; social frontiers (Dean, et al. 2018)



Why might we want to use maps in our data visualisation?

Argument 2: Because it's useful.

- Knowing the spatial characteristics of a societal phenomenon or problem (how concentrated or dispersed it is) can help in designing effective solutions. For example:
  - Targeted mailing to residences in certain areas about subsidised insulation programmes.
  - Identifying potentially effective areas for place-based intervention, e.g. drug harm reduction facilities in areas with high rates of drug overdose.
  - Identifying places with poor access to the internet to help close the digital divide.

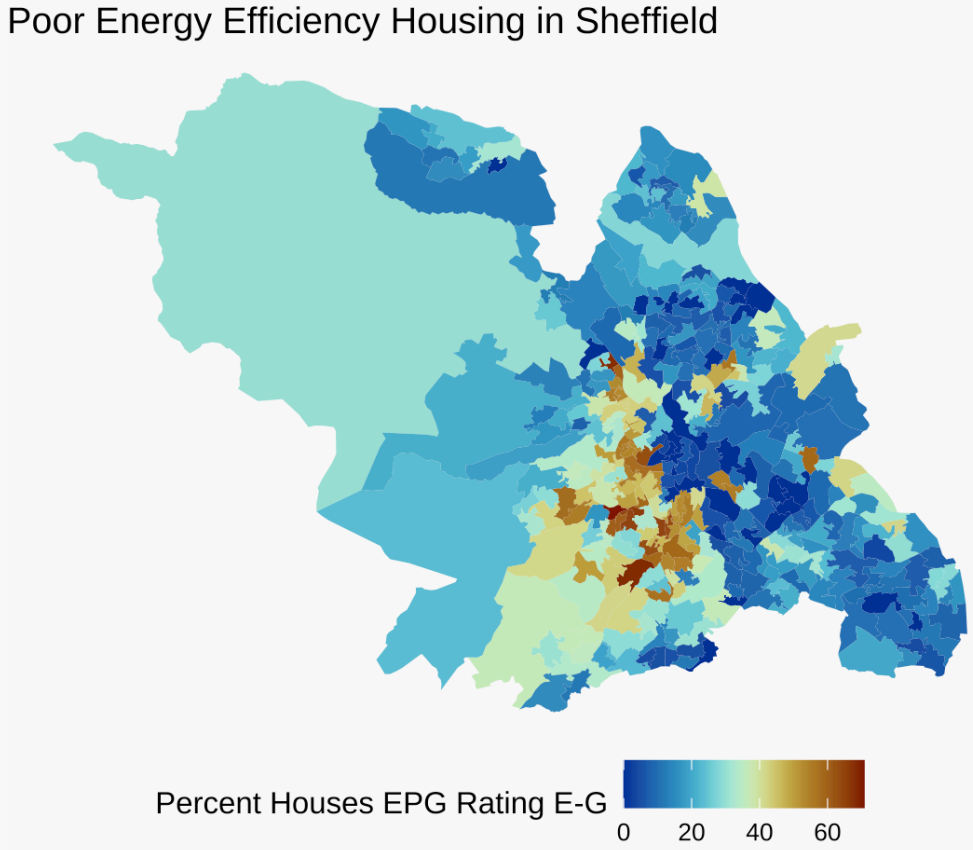


Why might we want to use maps in our data visualisation?

Argument 3: Because why something happens can often be related to *where it is*.

> "Everything is related to everything else, but near things are more related than distant things."
>
> Tobler's first law of geography. Miller (2004)

- We might *think* two things are related because they often happen together, e.g. bad air quality ('miasma') and illness (such as cholera).
- Actually, they may only be *spuriously* related *because they share the same space*, e.g. the Broad Street water pump.



Week 8: Maps — Part II

Spatial Visualisation in R: Reading in spatial data (boundaries and centroids)


sf:
SPATIAL DATA ... SIMPLIFIED.
ATTRIBUTES + Geometries
1 Y 2.6 blue
6 N 5.0 red
12 Y 11.9 green
Y 13.8 blue
N 21.7 red
GLUE
tape
Sticky geometries: for people who love their maps and sanity.
Horst '18

Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

- Comes in multiple different **formats** (e.g. Shapefiles, GeoJSON, GeoPackage)

Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

- Comes in multiple different **formats** (e.g. Shapefiles, GeoJSON, GeoPackage)
- Can be **boundaries** or **points**.

Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

- Comes in multiple different **formats** (e.g. Shapefiles, GeoJSON, GeoPackage)
- Can be **boundaries** or **points**.
- Uses different forms of **projection** and coordinate reference systems.

Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

- Comes in multiple different **formats** (e.g. Shapefiles, GeoJSON, GeoPackage)
- Can be **boundaries** or **points**.
- Uses different forms of **projection** and coordinate reference systems.
- Comes in different forms of **geometry type** and coordinates systems (e.g. X & Y coordinates, latitude & longitude coordinates)

Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

- Comes in multiple different **formats** (e.g. Shapefiles, GeoJSON, GeoPackage)
- Can be **boundaries** or **points**.
- Uses different forms of **projection** and coordinate reference systems.
- Comes in different forms of **geometry type** and coordinates systems (e.g. X & Y coordinates, latitude & longitude coordinates)
- Some packages only work with certain combinations of the above.

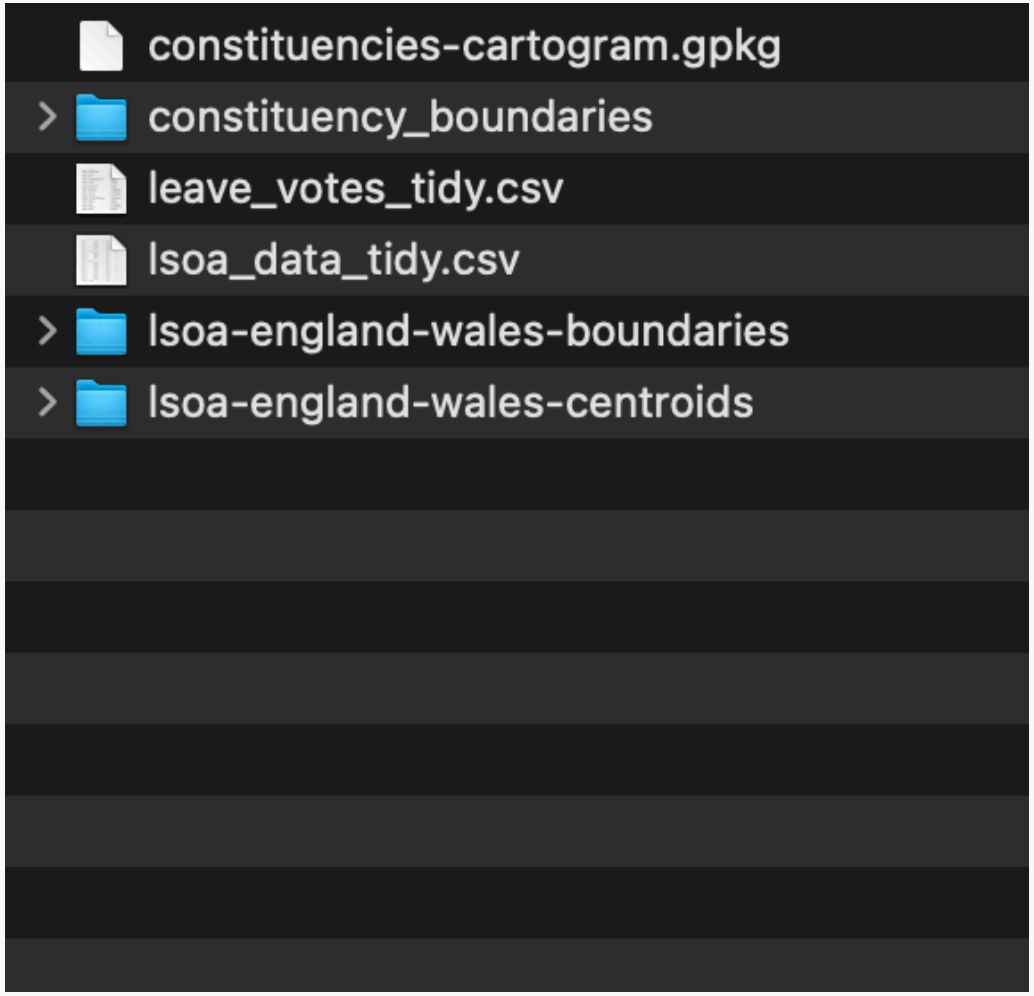
Spatial data is... complicated.

We saw a bit of this last week; let's break down what we did at the start a bit.

- Comes in multiple different **formats** (e.g. Shapefiles, GeoJSON, GeoPackage)
- Can be **boundaries** or **points**.
- Uses different forms of **projection** and coordinate reference systems.
- Comes in different forms of **geometry type** and coordinates systems (e.g. X & Y coordinates, latitude & longitude coordinates)
- Some packages only work with certain combinations of the above.
- However, the `sf` package makes reading in and converting data to different types very easy.

Reading in spatial data

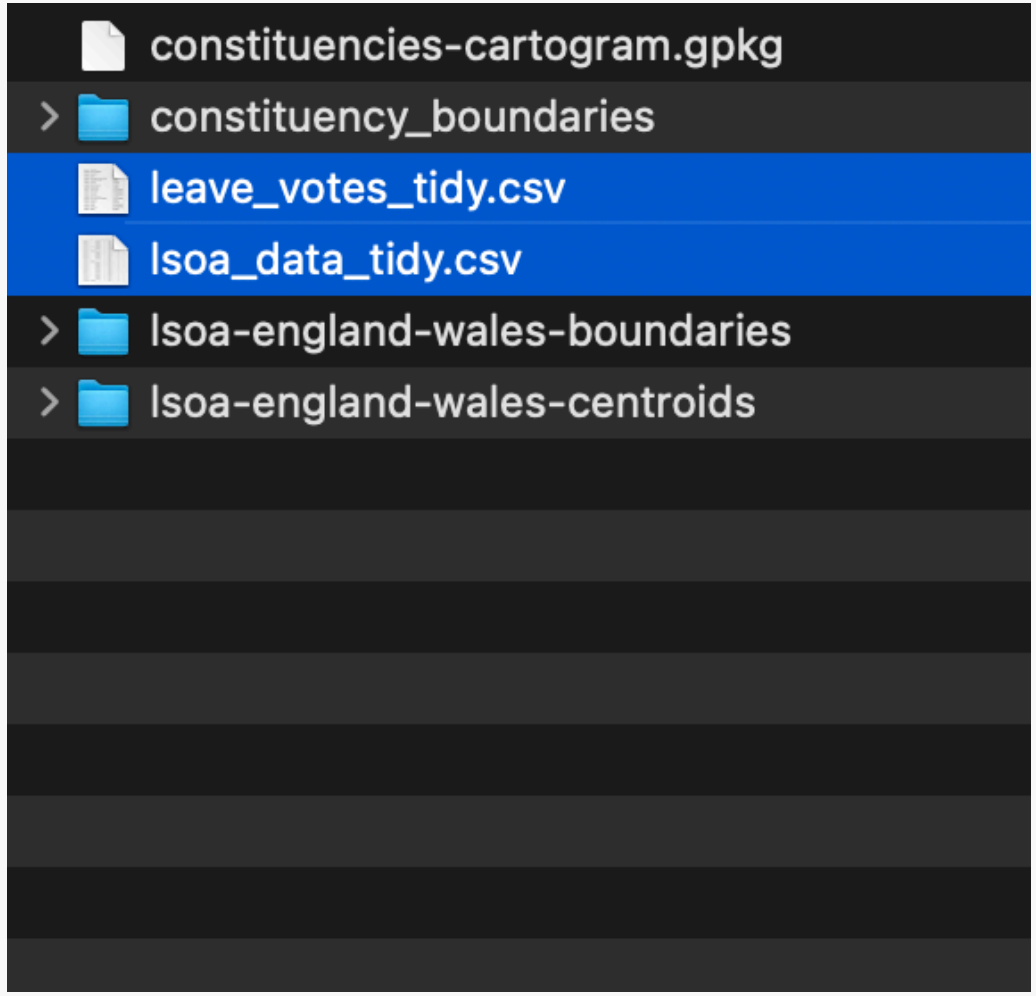
On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.



Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

- These two files are just plain CSVs (comma-separated values). They contain my non-spatial data.
- These might be things I want to project onto the spatial boundaries (e.g. EU leave vote share by constituency; percentage of houses with E to G energy efficiency)



Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

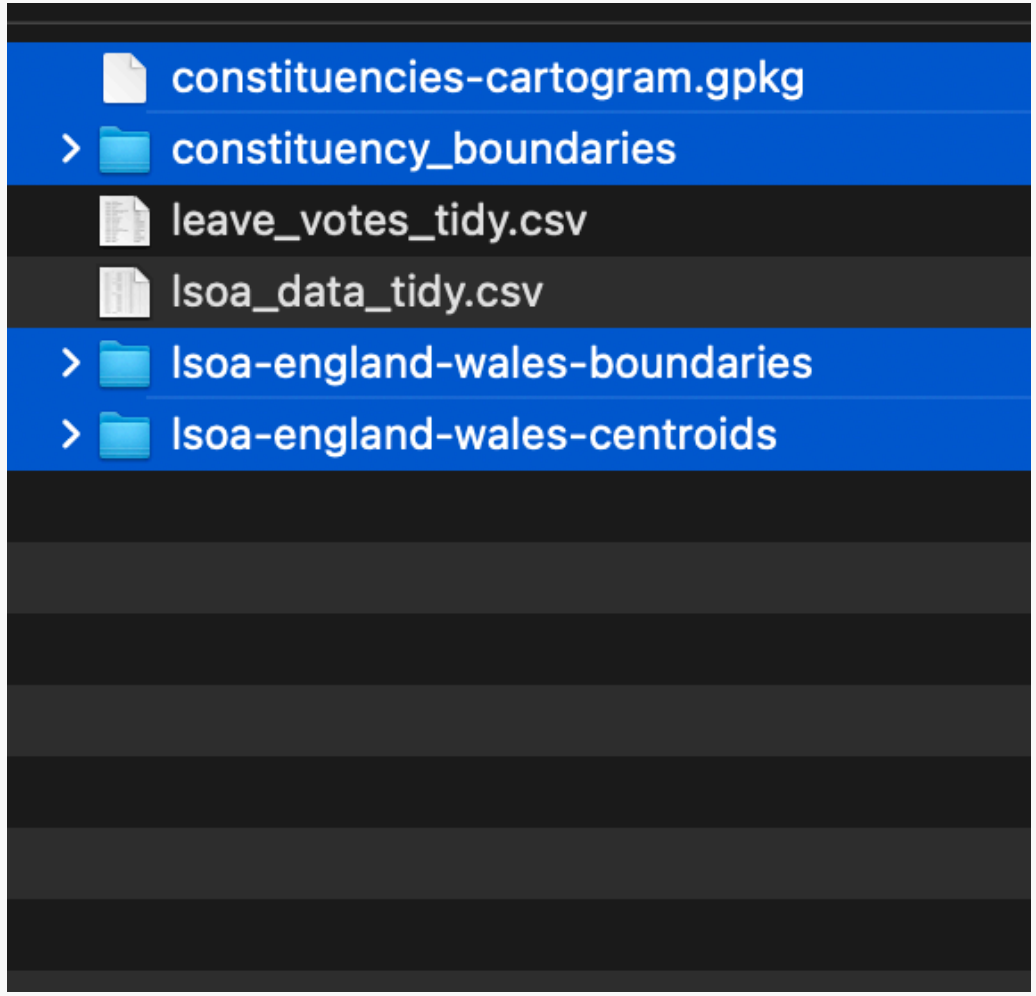
- These two files are just plain CSVs (comma-separated values). They contain my non-spatial data.
- These might be things I want to project onto the spatial boundaries (e.g. EU leave vote share by constituency; percentage of houses with E to G energy efficiency)
- Inportantly, they contain a **unique identifier** for each geographic area. For example, in the `leave_votes_tidy.csv` file, there is a column called `const_cd`, or "constituency code". This will allow me to join it to the spatial data later.

```
# A tibble: 650 × 3
   const_cd  constituency
   <chr>     <chr>
 1 E14000582 Boston and Skegness
 2 E14001011 Walsall North
 3 E14000642 Clacton
 4 E14000933 South Basildon and East
 5 E14000771 Kingston upon Hull East
 6 E14000622 Castle Point
 7 E14000973 Stoke-on-Trent North
 8 E14000669 Doncaster North
 9 E14000717 Great Yarmouth
10 E14000716 Great Grimsby
# … with 640 more rows
```

Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

- I also have two different types of spatial data: a GeoPackage dataset `constituencies-cartogram.gpkg`, and several shapefile data folders `constituency_boundaries`, `lsoa-england-wales-boundaries`, and `lsoa-england-wales-centroids`.



Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

- GeoPackages contain several different 'layers' of spatial data. When you read them in, you need to specify which layer you want. Often, you can combine several layers to get a desired effect (e.g. one layer for the entire country, another for the cities, another for small areas)
    - You can view the layers in a GeoPackage type data file using the `st_layers` function in `sf`

```
library(sf)

st_layers("data/constituencies-cartogram.gpkg")
```

```
## Driver: GPKG
## Available layers:
##         layer_name geometry_type features fields
## 1    1 Group names          Point       59      4 OSGB36 / Br
## 2 2 Group outlines Multi Polygon       60      3 OSGB36 / Br
## 3  3 City outlines Multi Polygon       55      4 OSGB36 / Br
## 4 4 Constituencies Multi Polygon      650      6 OSGB36 / Br
## 5     5 Background Multi Polygon        3      2 OSGB36 / Br
## 6     layer_styles             NA        5     12
```

Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

- GeoPackages contain several different 'layers' of spatial data. When you read them in, you need to specify which layer you want. Often, you can combine several layers to get a desired effect (e.g. one layer for the entire country, another for the cities, another for small areas)
    - You can view the layers in a GeoPackage type data file using the `st_layers` function in `sf`
    - We can see that the package has multiple layers, e.g. "3 City outlines" contains the spatial data required to draw larger city groups, whereas "4 Constituencies" contains the spatial data required to draw parliamentary constituencies.

```
library(sf)

st_layers("data/constituencies-cartogram.gpkg")
```

```
## Driver: GPKG
## Available layers:
##         layer_name geometry_type features fields
## 1    1 Group names          Point       59      4 OSGB36 / Br
## 2 2 Group outlines Multi Polygon       60      3 OSGB36 / Br
## 3  3 City outlines Multi Polygon       55      4 OSGB36 / Br
## 4 4 Constituencies Multi Polygon      650      6 OSGB36 / Br
## 5     5 Background Multi Polygon        3      2 OSGB36 / Br
## 6     layer_styles             NA        5     12
```

Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

- Once we've decided which layer or layers we want, we can read them into `R` using the `st_read` function.

```
wpc_background <- st_read("data/constituencies-cartogram.gpkg",
                          layer = "5 Background")
```

```
## Reading layer `5 Background' from data source
##   `/Users/calumwebb/Library/Mobile Documents/com~apple~Cloudl
##   using driver `GPKG'
## Simple feature collection with 3 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 5.987083 ymin: 1.761197 xmax: 56.83708
## Projected CRS: OSGB36 / British National Grid
```

```
wpc_constituencies <- st_read("data/constituencies-cartogram.gp
                          layer = "4 Constituencies")
```

```
## Reading layer `4 Constituencies' from data source
##   `/Users/calumwebb/Library/Mobile Documents/com~apple~Cloudl
##   using driver `GPKG'
## Simple feature collection with 650 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 7.337122 ymin: 2.106932 xmax: 55.78812
## Projected CRS: OSGB36 / British National Grid
```

Reading in spatial data

On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.

- Shapefiles are a little bit easier to work with. We only need to use `st_read` and tell it where the shape file is.
    - You can find many geographical boundary and centroid shapefiles for UK areas at https://geoportal.statistics.gov.uk

```
lsoa_bounds <- st_read("data/lsoa-england-wales-boundaries/")
```

```
## Reading layer `Lower_Layer_Super_Output_Areas_December_2011_
##   using driver `ESRI Shapefile'
## Simple feature collection with 34753 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -6.418524 ymin: 49.86474 xmax: 1.762942
## Geodetic CRS:  WGS 84
```

```
lsoa_centres <- st_read("data/lsoa-england-wales-centroids/")
```

```
## Reading layer `Lower_Layer_Super_Output_Areas_(December_2011
##   using driver `ESRI Shapefile'
## Simple feature collection with 34753 features and 3 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 90590.6 ymin: 10638.17 xmax: 655020.5 y
## Projected CRS: OSGB36 / British National Grid
```

Week 8: Maps — Part III

Spatial Analysis in R: Joining other data to spatial data

Joining other data to spatial data

```
lsoa_boundaries
```

```
## Simple feature collection with 34753 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -6.418524 ymin: 49.86474 xmax: 1.762942 ymax: 55.81107
## Geodetic CRS:  WGS 84
## # A tibble: 34,753 × 7
##    objectid lsoa11cd  lsoa11nm          lsoa11nmw st_areasha
##       <dbl> <chr>     <chr>             <chr>          <dbl>
##  1        1 E01000001 City of London 0… City of …    133321.
##  2        2 E01000002 City of London 0… City of …    226191.
##  3        3 E01000003 City of London 0… City of …     57303.
##  4        4 E01000005 City of London 0… City of …    190739.
##  5        5 E01000006 Barking and Dage… Barking …    144196.
##  6        6 E01000007 Barking and Dage… Barking …    198135.
##  7        7 E01000008 Barking and Dage… Barking …    193425.
##  8        8 E01000009 Barking and Dage… Barking …    128592.
##  9        9 E01000010 Barking and Dage… Barking …    348848.
## 10       10 E01000011 Barking and Dage… Barking …     90298.
## # ℹ 34,743 more rows
## # ℹ 2 more variables: st_lengths <dbl>,
## #   geometry <MULTIPOLYGON [°]>
```

Once we've read in our spatial data we normally need to add the things we're interested in mapping or analysing spatially to it. Spatial data doesn't often come with social science data attached (and vice versa).

We can merge datasets together using the `join` functions in the `tidyverse` package. Specifically, `left_join`.

Joining other data to spatial data

Let's read in some non-spatial, social science-related LSOA level data.

```
lsoa_data <- read_csv("data/lsoa_data_tidy.csv")
lsoa_data
```

```
## # A tibble: 32,844 × 10
##    lsoa_code lsoa_name idaopi_2019 percent_eg_rated
##    <chr>     <chr>           <dbl>            <dbl>
##  1 E01031349 Adur 001A         8.4              32.5
##  2 E01031350 Adur 001B        21.8              32.1
##  3 E01031351 Adur 001C         8.1              30.6
##  4 E01031352 Adur 001D         8.5              38.6
##  5 E01031370 Adur 001E        11.2              21.5
##  6 E01031374 Adur 001F         9.9              15.4
##  7 E01031338 Adur 002A         7.3              22.5
##  8 E01031339 Adur 002B         6.6              13.8
##  9 E01031340 Adur 002C         4.2              28.2
## 10 E01031365 Adur 002D        12.4              36.6
## # ℹ 32,834 more rows
## # ℹ 6 more variables: housesales_1998 <dbl>,
## #   housesales_2018 <dbl>, utla17nm <chr>,
## #   housesales_1998_quantiles <dbl>,
## #   housesales_2018_quantiles <dbl>, population <dbl>
```

Joining other data to spatial data

Let's read in some non-spatial, social science-related LSOA level data.

```
lsoa_data <- read_csv("data/lsoa_data_tidy.csv")
lsoa_data
```

```
## # A tibble: 32,844 × 10
##    lsoa_code lsoa_name idaopi_2019 percent_eg_rated
##    <chr>     <chr>           <dbl>            <dbl>
##  1 E01031349 Adur 001A         8.4             32.5
##  2 E01031350 Adur 001B        21.8             32.1
##  3 E01031351 Adur 001C         8.1             30.6
##  4 E01031352 Adur 001D         8.5             38.6
##  5 E01031370 Adur 001E        11.2             21.5
##  6 E01031374 Adur 001F         9.9             15.4
##  7 E01031338 Adur 002A         7.3             22.5
##  8 E01031339 Adur 002B         6.6             13.8
##  9 E01031340 Adur 002C         4.2             28.2
## 10 E01031365 Adur 002D        12.4             36.6
## # ℹ 32,834 more rows
## # ℹ 6 more variables: housesales_1998 <dbl>,
## #   housesales_2018 <dbl>, utla17nm <chr>,
## #   housesales_1998_quantiles <dbl>,
## #   housesales_2018_quantiles <dbl>, population <dbl>
```

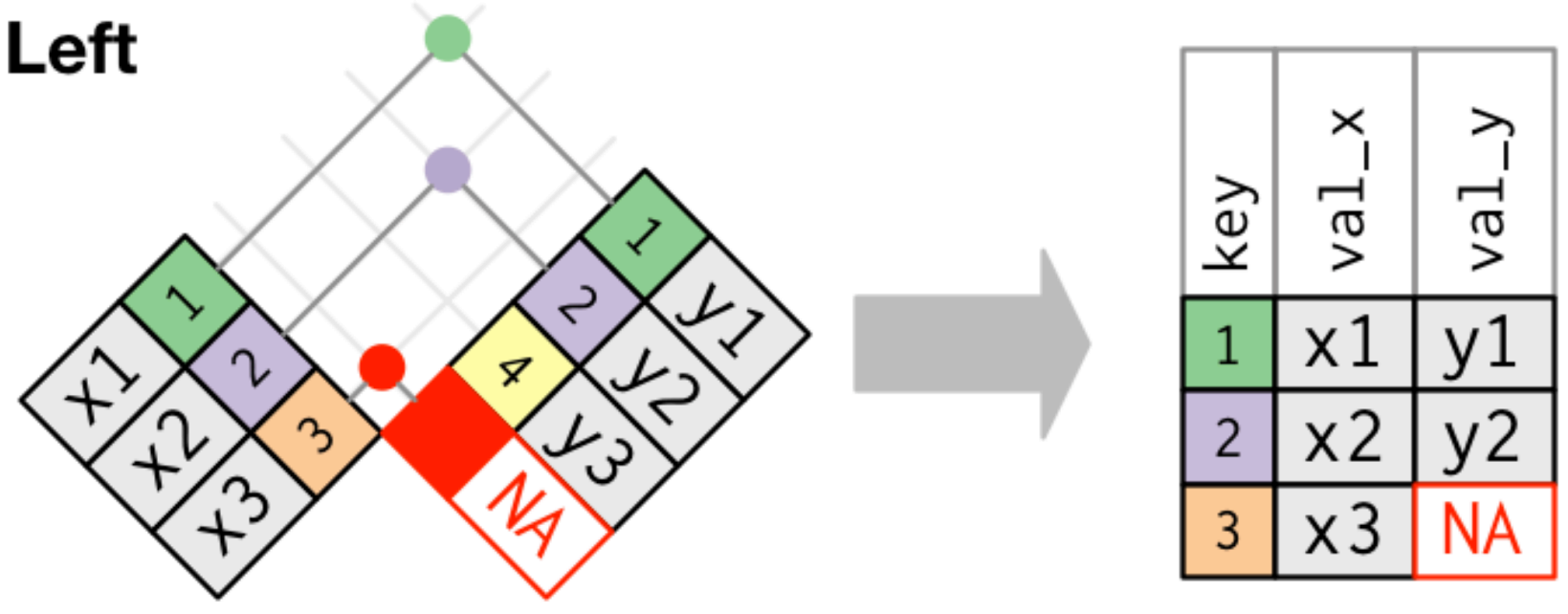
```
lsoa_boundaries
```

```
## Simple feature collection with 34753 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -6.418524 ymin: 49.86474 xmax: 1.762942
## Geodetic CRS:  WGS 84
## # A tibble: 34,753 × 7
##    objectid lsoa11cd  lsoa11nm           lsoa11nmw st_areasha
##       <dbl> <chr>     <chr>              <chr>          <dbl>
##  1        1 E01000001 City of London 0… City of …    133321.
##  2        2 E01000002 City of London 0… City of …    226191.
##  3        3 E01000003 City of London 0… City of …     57303.
##  4        4 E01000005 City of London 0… City of …    190739.
##  5        5 E01000006 Barking and Dage… Barking …    144196.
##  6        6 E01000007 Barking and Dage… Barking …    198135.
##  7        7 E01000008 Barking and Dage… Barking …    193425.
##  8        8 E01000009 Barking and Dage… Barking …    128592.
##  9        9 E01000010 Barking and Dage… Barking …    348848.
## 10       10 E01000011 Barking and Dage… Barking …     90298.
## # ℹ 34,743 more rows
## # ℹ 2 more variables: st_lengths <dbl>,
## #   geometry <MULTIPOLYGON [°]>
```

Note the same ID column for small areas: `lsoa_code` in the `lsoa_data` object, `lsoa11cd` in the `lsoa_boundaries` object

Joining other data to spatial data

`left_join` (Wickham & Grolemund, 2017)



Joining other data to spatial data

What would a `left_join` of the following two datasets look like?

data_age

ID	age
10012	60
10013	43
10014	22
10015	56

data_smoking

ID	smoker
10012	Yes
10013	No
10015	No
10016	Yes

left_join(data_age, data_smoking, by = "ID")

ID	age	smoker

Joining other data to spatial data

What would a `left_join` of the following two datasets look like?

data_age

ID	age
10012	60
10013	43
10014	22
10015	56

data_smoking

ID	smoker
10012	Yes
10013	No
10015	No
10016	Yes

left_join(data_age, data_smoking, by = "ID")

ID	age	smoker
10012	60	Yes
10013	43	No
10014	22	NA
10015	56	No

Joining other data to spatial data

`left_join` takes three main arguments: `x` (your 'left' dataset), `y` (your 'right' dataset), and `by`, which identifies the key (or ID) column.

The `by` argument is always entered as a string. **Important:** If the name of the key/ID variable is different in the two datasets, the `by` argument needs to be written like: `by = c("left_key_variable" = "right_key_variable")`

Our spatial data should **always** be our "left" variable, else we'll lose our spatial information.

```
lsoa_boundaries_c <- left_join(lsoa_boundaries, lsoa_data, by = c("lsoa11cd" = "lsoa_cod
lsoa_boundaries_c
```

```
## Simple feature collection with 34753 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -6.418524 ymin: 49.86474 xmax: 1.762942 ymax: 55.81107
## Geodetic CRS:  WGS 84
## # A tibble: 34,753 × 16
##    objectid lsoa11cd  lsoa11nm                 lsoa11nmw  st_areasha st_lengths
##       <dbl> <chr>     <chr>                    <chr>           <dbl>      <dbl>
##  1        1 E01000001 City of London 001A      City of L…    133321.      2292.
##  2        2 E01000002 City of London 001B      City of L…    226191.      2434.
##  3        3 E01000003 City of London 001C      City of L…     57303.      1142.
##  4        4 E01000005 City of London 001E      City of L…    190739.      2168.
##  5        5 E01000006 Barking and Dagenham 016A Barking a…    144196.      1936.
##  6        6 E01000007 Barking and Dagenham 015A Barking a…    198135.      2824.
##  7        7 E01000008 Barking and Dagenham 015B Barking a…    193425.      3908.
##  8        8 E01000009 Barking and Dagenham 016B Barking a…    128592.      2066.
##  9        9 E01000010 Barking and Dagenham 015C Barking a…    348848.      3098.
## 10       10 E01000011 Barking and Dagenham 016C Barking a…     90298.      1496.
## # ℹ 34,743 more rows
## # ℹ 10 more variables: geometry <MULTIPOLYGON [°]>, lsoa_name <chr>,
## #   idaopi_2019 <dbl>, percent_eg_rated <dbl>, housesales_1998 <dbl>,
## #   housesales_2018 <dbl>, utla17nm <chr>, housesales_1998_quantiles <dbl>,
## #   housesales_2018_quantiles <dbl>, population <dbl>
```

Week 8: Maps — Part IV

Spatial Data Visualisation — Choropleth Maps

Choropleth Maps

Okay, here's a recap of what we did in the workshop this week.

Let's start by just plotting our LSOA boundaries. But first, I'm going to select just Sheffield by filtering out all data that doesn't have "Sheffield" in the LSOA name using `str_detect`.

I'm also going to re-create the housing quantiles variables so that they now reflect only the most popular buying areas of Sheffield.

```
sheffield_data <- lsoa_boundaries_c %>%
  filter(str_detect(lsoa11nm, "Sheffield")) %>%
  mutate(housesales_1998_quantiles = ntile(housesales_1998, 20),
         housesales_2018_quantiles = ntile(housesales_2018, 20))


sheffield_data
```

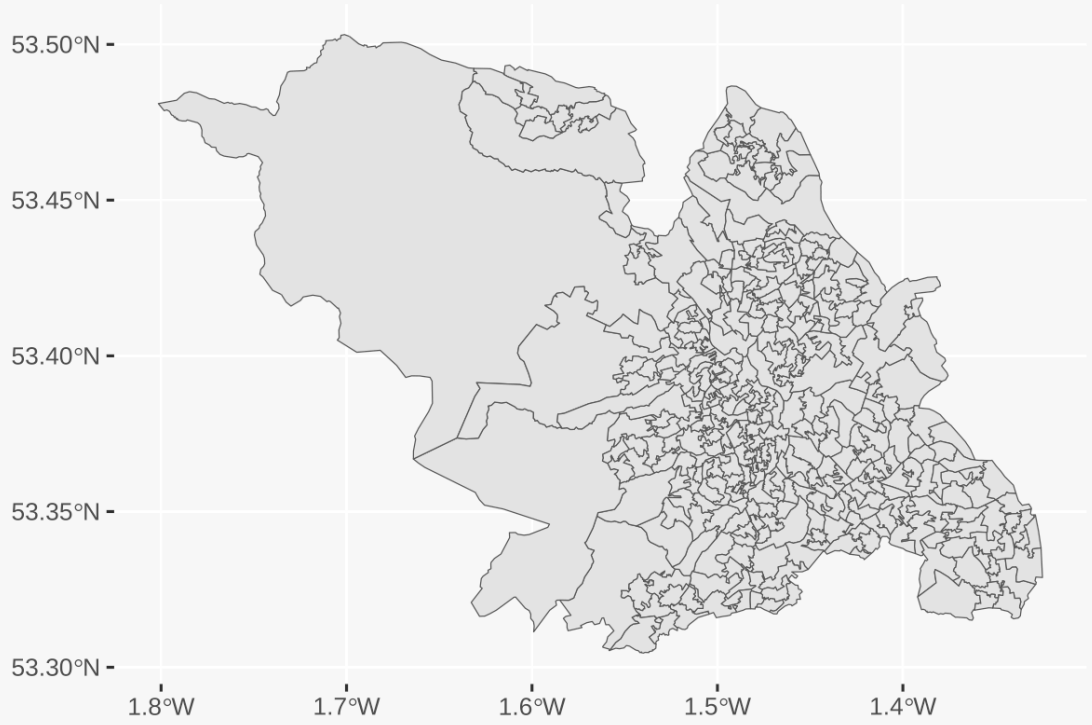
```
## Simple feature collection with 345 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1.801471 ymin: 53.30457 xmax: -1.324727 ymax: 53.50314
## Geodetic CRS:  WGS 84
## # A tibble: 345 × 16
##    objectid lsoa11cd  lsoa11nm       lsoa11nmw      st_areasha st_lengths
##  *    <dbl> <chr>     <chr>          <chr>               <dbl>      <dbl>
##  1     7625 E01007823 Sheffield 069A Sheffield 069A    421798.      5358.
##  2     7626 E01007824 Sheffield 066A Sheffield 066A    345346.      3627.
##  3     7627 E01007825 Sheffield 066B Sheffield 066B   1264119.      6425.
##  4     7628 E01007826 Sheffield 066C Sheffield 066C   1051343.      6100.
##  5     7629 E01007827 Sheffield 064A Sheffield 064A   1541137.      7910.
##  6     7630 E01007828 Sheffield 059A Sheffield 059A    280466.      3537.
##  7     7631 E01007829 Sheffield 059B Sheffield 059B    443268.      3539.
##  8     7632 E01007830 Sheffield 064B Sheffield 064B    290064.      3536.
##  9     7633 E01007831 Sheffield 059C Sheffield 059C    307751.      4402.
## 10     7634 E01007832 Sheffield 059D Sheffield 059D    200169.      2267.
## # ℹ 335 more rows
## # ℹ 10 more variables: geometry <MULTIPOLYGON [°]>, lsoa_name <chr>,
```

Choropleth Maps

Now let's plot our map using `ggplot`.

To plot spatial features (or 'simple features'), we use `geom_sf` from the `sf` package.

```
ggplot(data = sheffield_data) +
  geom_sf()
```

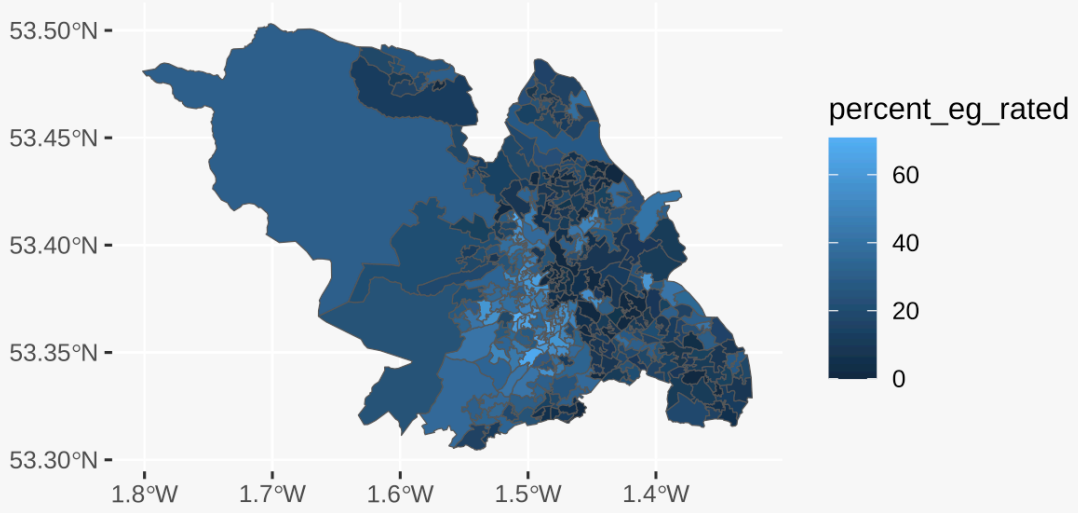


Choropleth Maps

Now let's plot our map using `ggplot`.

We can 'fill' our shapes in using variables in the dataset.

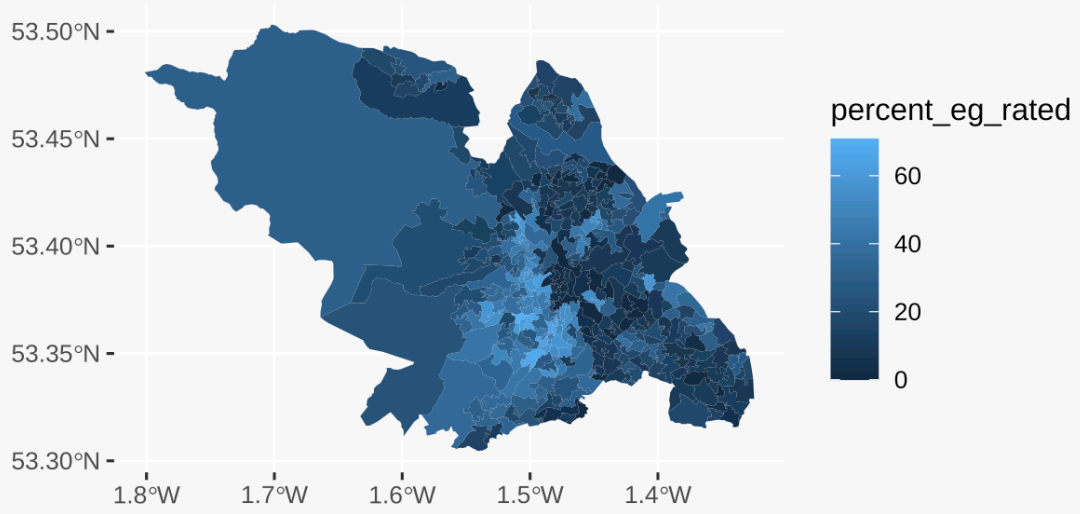
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated))
```



Choropleth Maps

Let's see if we can improve how our map looks with a few extra options. Let's start by turning off the boundary lines with `colour = "transparent"`.

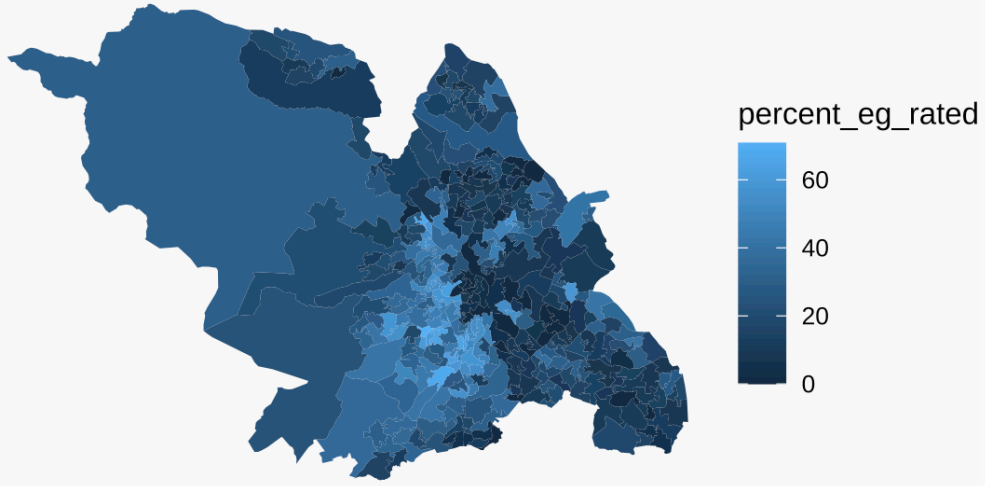
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent")
```



Choropleth Maps

That's better. Let's get rid of the plot background as we don't really need to know the latitude and longitude. We can use `theme_void()` to do this quickly.

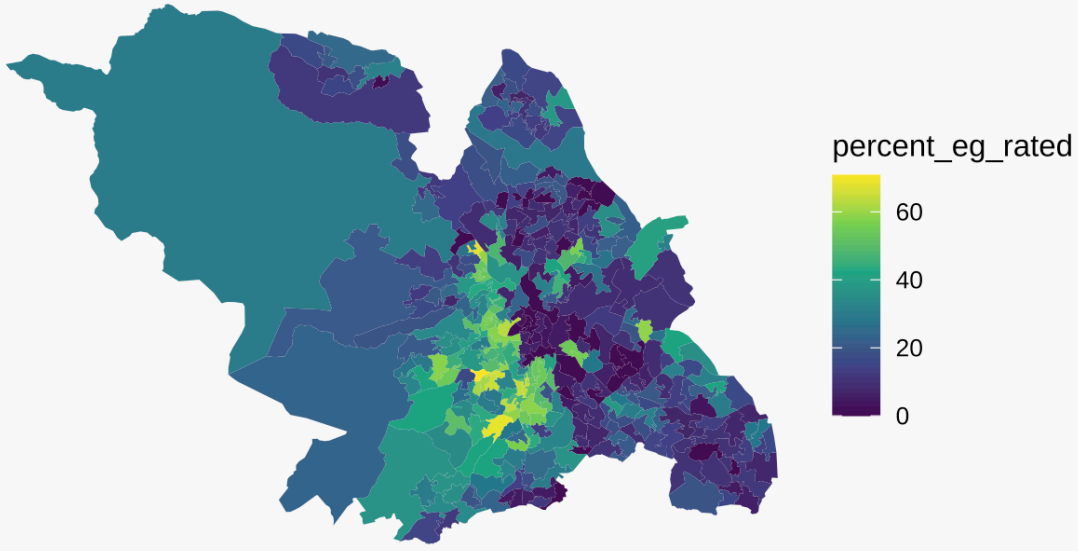
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent") +
  theme_void()
```



Choropleth Maps

We can also change our colour scheme if we want. One easy option is to use `scale_fill_viridis_c()`

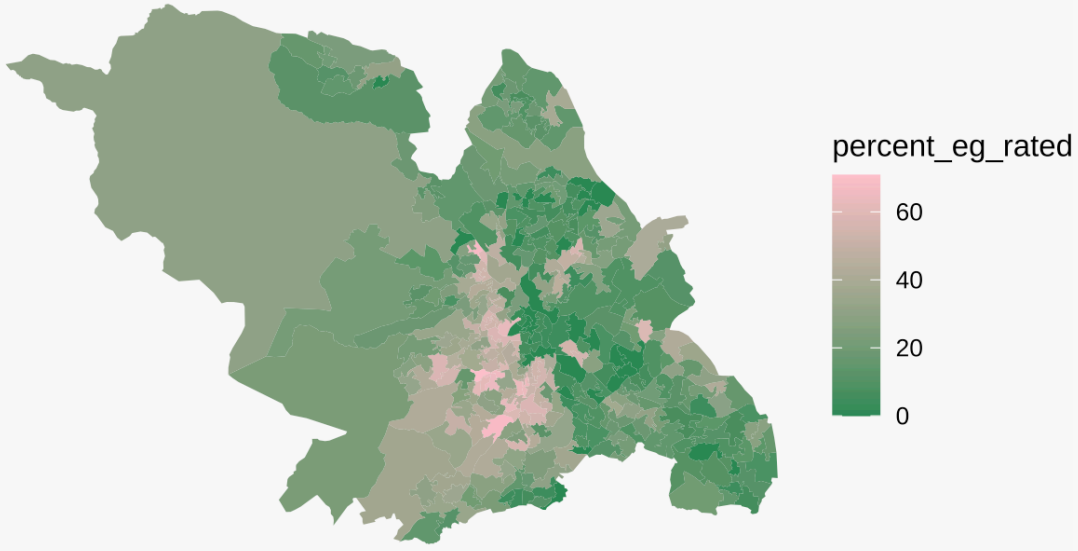
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent") +
  theme_void() +
  scale_fill_viridis_c()
```



Choropleth Maps

We can choose our own high and low colours using the `scale_fill_gradient()` function if we want too.

```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent") +
  theme_void() +
  scale_fill_gradient(low = "seagreen", high = "pink")
```



What is a potential problem with our choropleth map?

Week 8: Maps — Part V

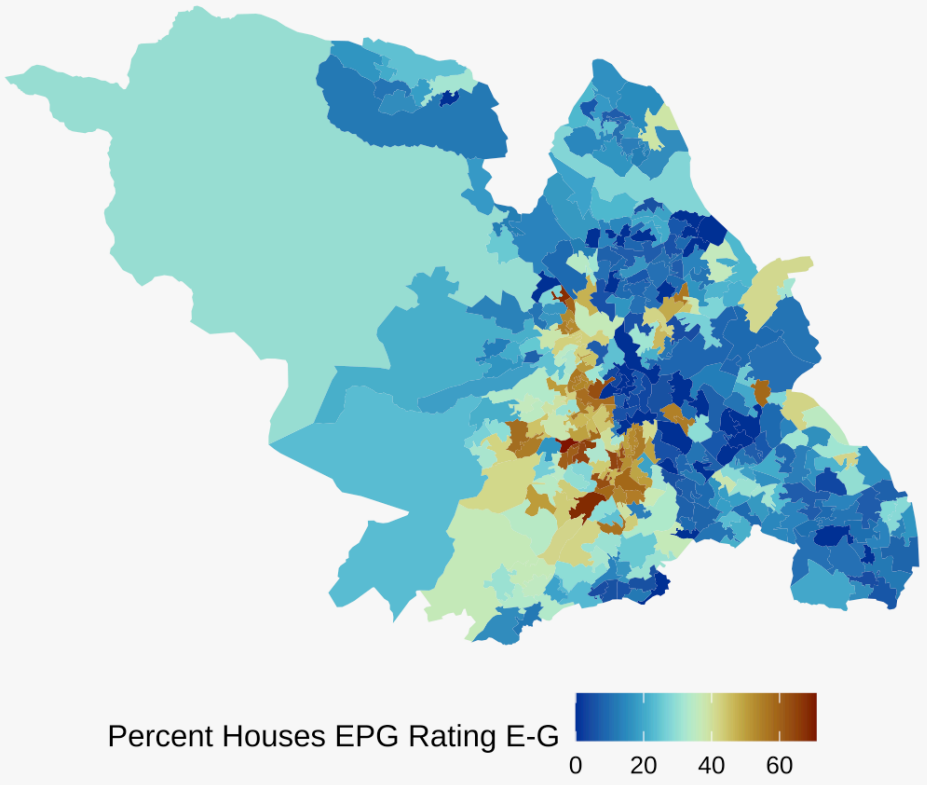
Spatial Data Visualisation — Cartograms

Cartograms

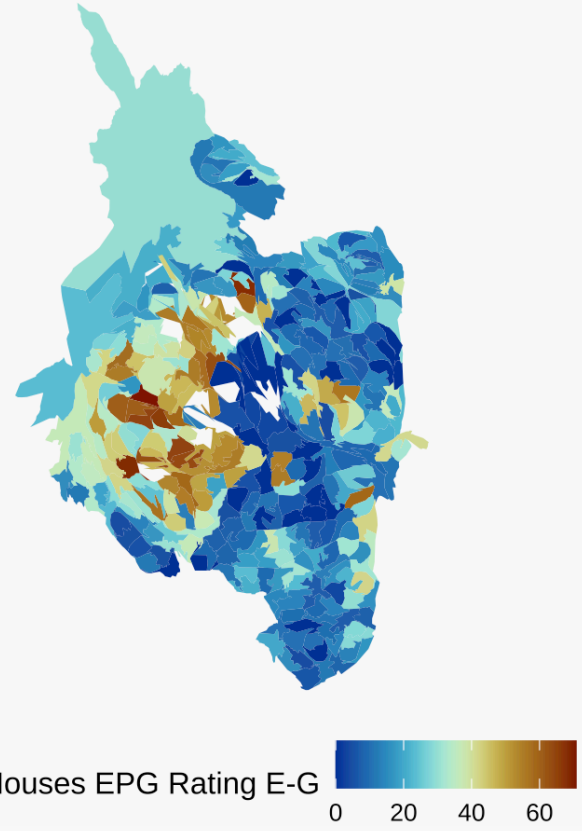
While a **choropleth map** refers to the colour-coding of a **geographically accurate** map, a **cartogram** refers to the colour-coding of a map that has been **transformed** in some way while (mostly) maintaining spatial arrangement, for example, to account for population size.

Cartograms

Choropleth

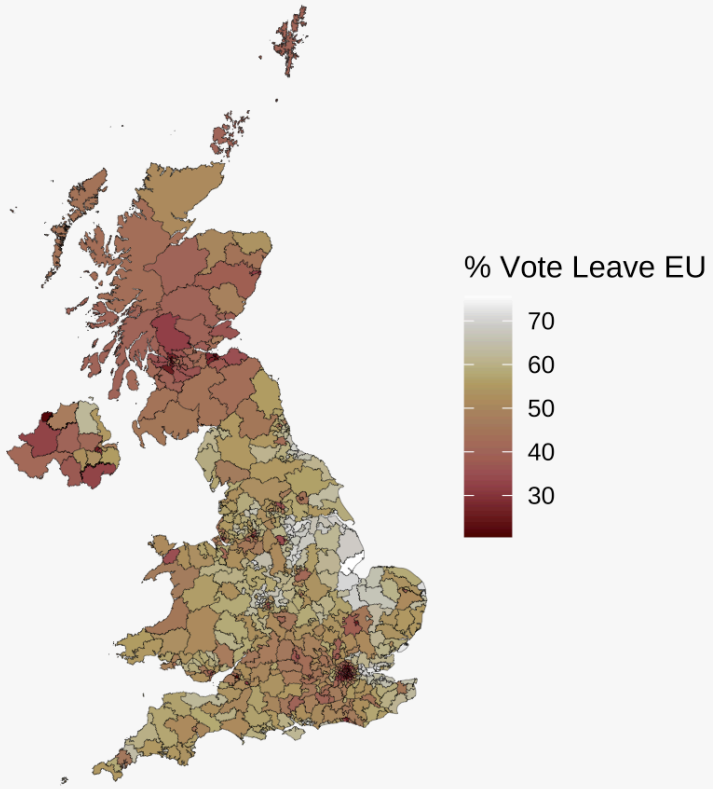


'Rubber Sheet' Population Cartogram

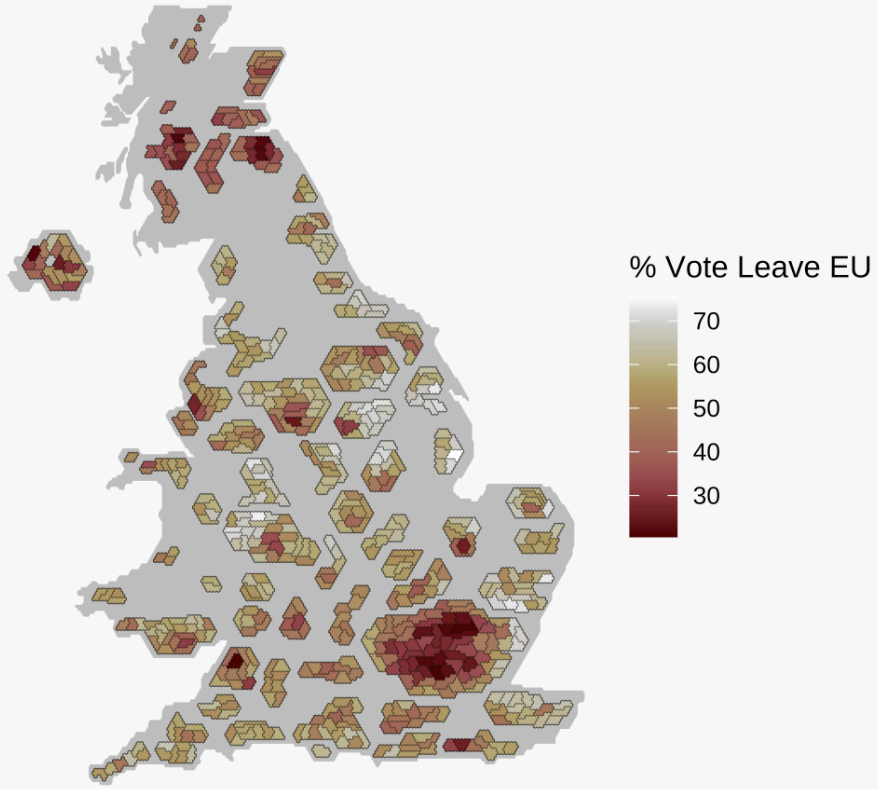


Cartograms

Choropleth



Non-contiguous Hex-map Cartogram



You can either find cartogram spatial data files that other people have already designed, for example, the one used in the previous slide is available from the House of Commons Library's github page.

You can either find cartogram spatial data files that other people have already designed, for example, the one used in the previous slide is available from the House of Commons Library's github page.

Or you can create them yourself using various cartogram algorithms — the `cartogram` package in `R` has been created to do this. For example, the above cartogram of Sheffield was created with the following code:

```
library(cartogram)
set.seed(2021)
# Need to convert the data to a projected spatial dataframe
# first
sheffield_longlat <- st_transform(sheffield_data, crs = 23038)
# Then can create a rubber sheet style population cartogram
# using the cartogram_cont function from cartogram
shef_cart <- cartogram_cont(sheffield_longlat,
                            weight = "population")
```

Week 8: Maps — Part VI

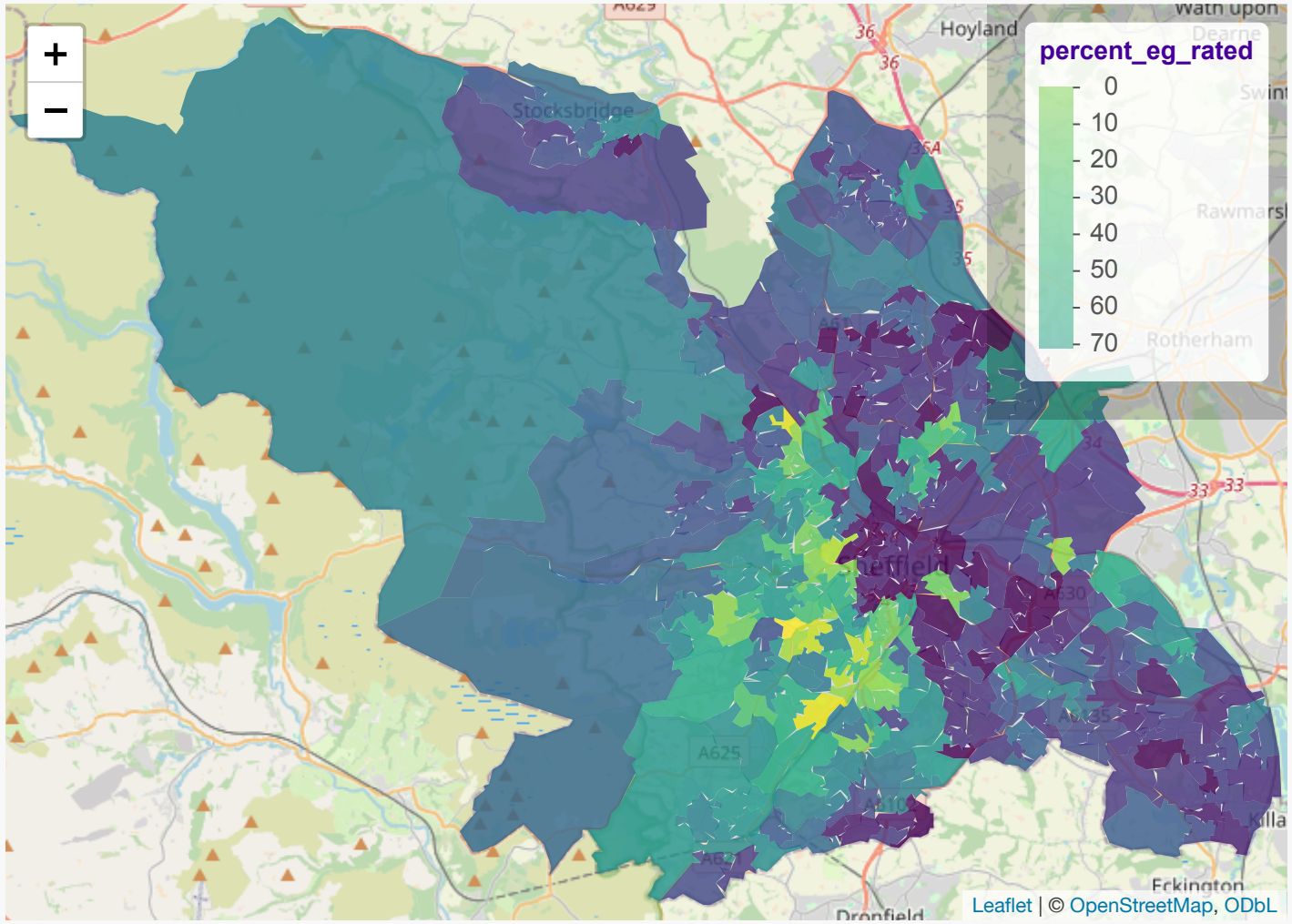
Spatial Data Visualisation — Interactive

Interactive Maps

Interactivity is increasingly used as a way to allow users to explore at various levels of scale.

One benefit to working in **R** is that it's relatively easy to create these impressive kinds of maps.

...It's still quite fiddly, but easier than you'd think using the `leaflet` package.



Interactive Maps

Interactivity is increasingly used as a way to allow researchers and others to explore at various levels of scale.

One benefit to working in **R** is that it's relatively easy to create these impressive kinds of maps.

...It's still quite fiddly, but easier than you'd think using the `leaflet` package.

Underlying Code

```
library(leaflet)

# Need to transform data into long and lat (rather than XY), as it's what leaflet
# uses
sheffield_data_longlat <- sheffield_data %>%
  st_transform(., crs = CRS("+proj=longlat +datum=WGS84"))

# Need to create a continuous colour palette for leaflet to use to fill in areas
sales_palette <- colorNumeric(palette = "viridis",
                              domain = sheffield_data_longlat$percent_eg_rated,
                              na.color = "transparent")

# Use the leaflet package in a similar way to ggplot
leaflet(sheffield_data_longlat) %>%
  addPolygons(weight = 0.1, stroke = FALSE,
              fillColor = ~sales_palette(percent_eg_rated),
              fillOpacity = 0.8) %>%
  addTiles() %>%
  addLegend(pal = sales_palette,
            values = ~percent_eg_rated)
```

Week 8: Maps — Part VI

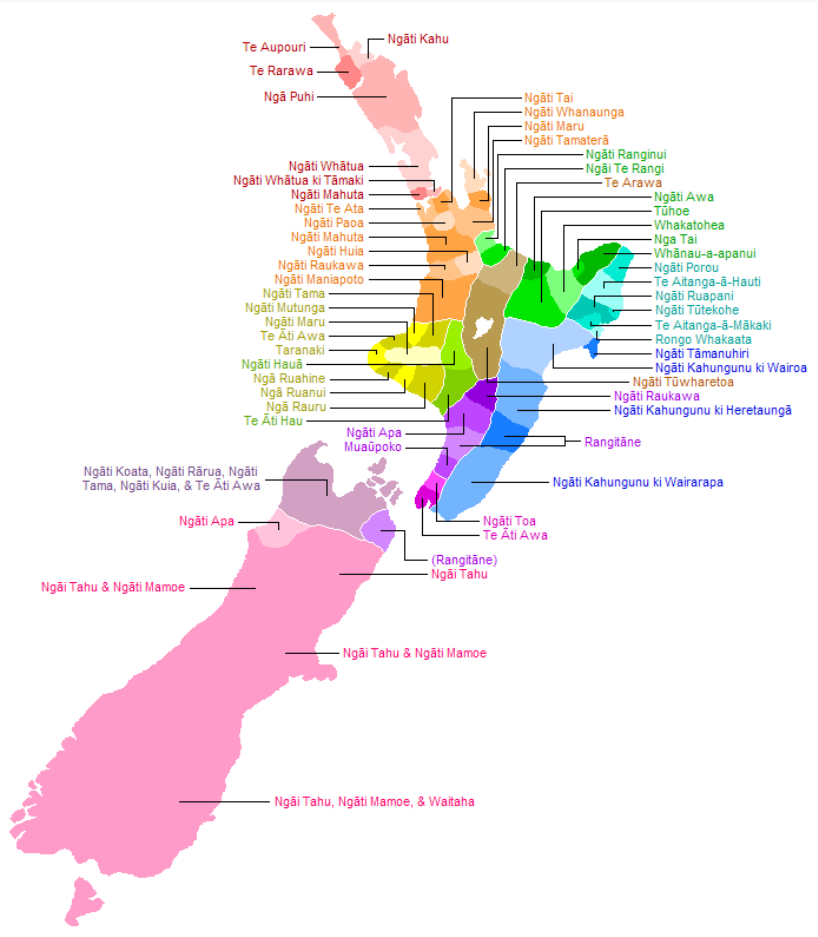
Critical awareness

Critical awareness

Map of Indigenous Territories overlaid on Settler territories in Canada

Critical awareness

Map of Pāhekā Colonial territories in Aotearoa New Zealand in 1858



Map of Māori Iwi in Aotearoa New Zealand

Critical awareness

Original Chinese Map of Minority Ethnic Groups in China

Translated Map of Minority Ethnic Groups in China

Source: University of Richmond

World Mercator projection with country going to true size

Summary

- Maps can be incredibly effective data visualisations because **people tend to care about and engage with *where they live* and *where they come from***. They can also reveal things that wouldn't be possible to identify without their spatial arrangement.

Summary

- Maps can be incredibly effective data visualisations because **people tend to care about and engage with *where they live* and *where they come from***. They can also reveal things that wouldn't be possible to identify without their spatial arrangement.
- However, spatial data is highly varied and **requires some technical skill to work with**. You need to be able to read in a variety of spatial data formats, convert them in various ways for use with packages of your choice, and join the geospatial data to the social science data of interest.

Summary

- Maps can be incredibly effective data visualisations because **people tend to care about and engage with *where they live* and *where they come from***. They can also reveal things that wouldn't be possible to identify without their spatial arrangement.
- However, spatial data is highly varied and **requires some technical skill to work with**. You need to be able to read in a variety of spatial data formats, convert them in various ways for use with packages of your choice, and join the geospatial data to the social science data of interest.
- We can represent spatial data visually through various types of maps: most commonly we use a static **choropleth**, but we may also wish to use **interactive maps** or **cartograms** to adjust for things like population size (remember: people vote, land doesn't vote!).

Summary

- Maps can be incredibly effective data visualisations because **people tend to care about and engage with *where they live* and *where they come from***. They can also reveal things that wouldn't be possible to identify without their spatial arrangement.
- However, spatial data is highly varied and **requires some technical skill to work with**. You need to be able to read in a variety of spatial data formats, convert them in various ways for use with packages of your choice, and join the geospatial data to the social science data of interest.
- We can represent spatial data visually through various types of maps: most commonly we use a static **choropleth**, but we may also wish to use **interactive maps** or **cartograms** to adjust for things like population size (remember: people vote, land doesn't vote!).
- **Maps are not objective representations of space**; they often reproduce borders with political and colonial origins and represent a specific notion of how land is divided and represented on a two dimensional plane.

Workshop this week

Let's draw some more maps. In particular, what we're going to focus on this week is combining different datasets together: what do you do if your map's contained in one file, but the relevant variable you want to use to illustrate the map is contained in another file that you need to download first?

TASKS TO LOOK AT IN ADVANCE OF WEEK 9:

Core tasks:

- **Before workshop**: Work through the task at the end of the workshop handout for this week