SMI606: Week 11
Spatial Analysis

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

Sign In

Learning Objectives

What will I learn? | How does this week fit into my course?

By the end of this week you will:

- Be able to identify when spatial analysis may be beneficial for a social research project/question.
- Be able to read spatial data into **R** for use in spatial analysis and visualisation.
- Be able to interpret and create choropleth and cartogram maps, showing how the variables of interest vary over space.
- Be able to calculate a statistic for measuring spatial concentration/dispersion — Moran's I — using **R**.
- Be aware of more advanced methods that can control for spatial autocorrelation in data.

Learning Objectives

What will I learn? | How does this week fit into my course?

- Spatial analysis is often a complementary component to any research project: it can serve as a way to identify interesting qualitative fieldwork sites or as a way to help communicate your research findings and create impact.
- Greater availability of computing power means that it's becoming easier (and more expected) to incorporate interactive features (like interactive maps) into research. In more advanced methods, it's also becoming increasingly common to account for and describe spatial autocorrelation.
- Greater prevalance of geo-tracking technology and sensors means there is likely to be growing amounts of interesting spatial data that can be analysed in the future.

Week 11: Spatial Analysis — Part I

What is spatial analysis and why learn it?

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**.

Work with a partner, or on your own, to document which areas (roughly defined) have grown or declined in popularity.

House Sales in 1998

House Sales in 2018

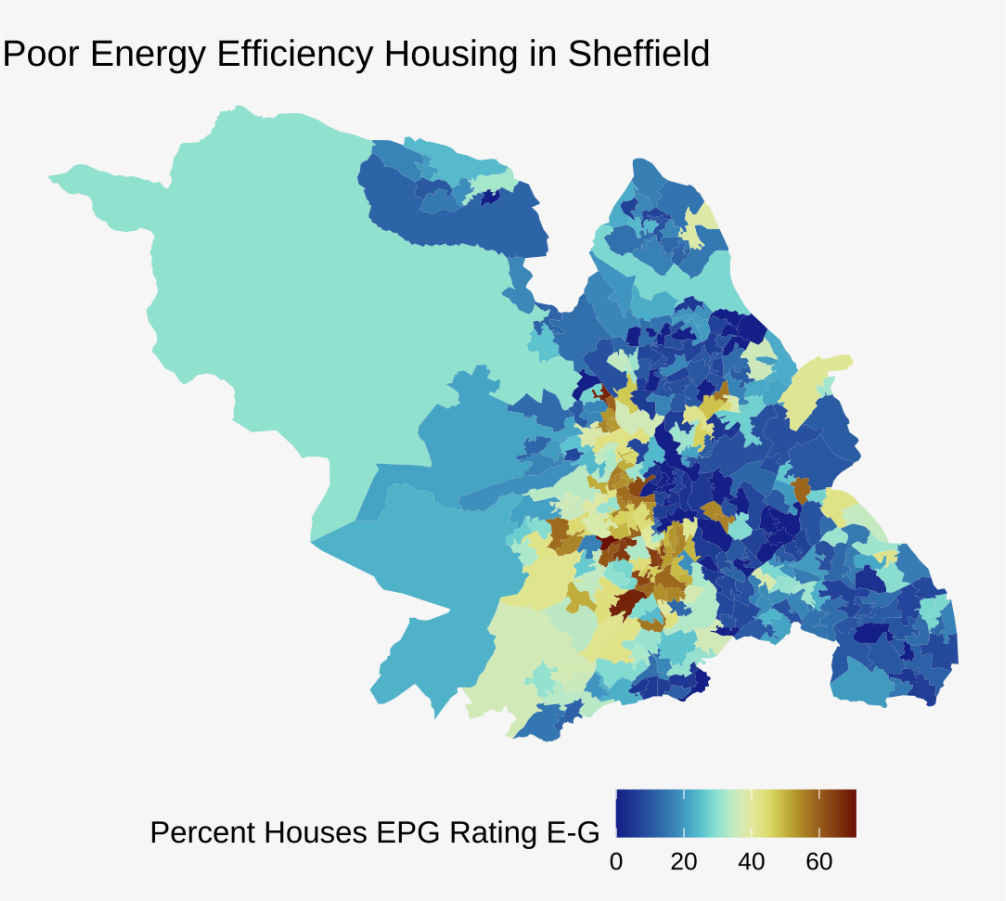
Put your findings on this Jamboard Page.

What is spatial analysis and why learn it?

Argument 1: Because it's interesting.

> If this is small area data... Why can't I just look at it on a map? That makes much more sense to me.
>
> My paraphrasing of many students' response to small area data

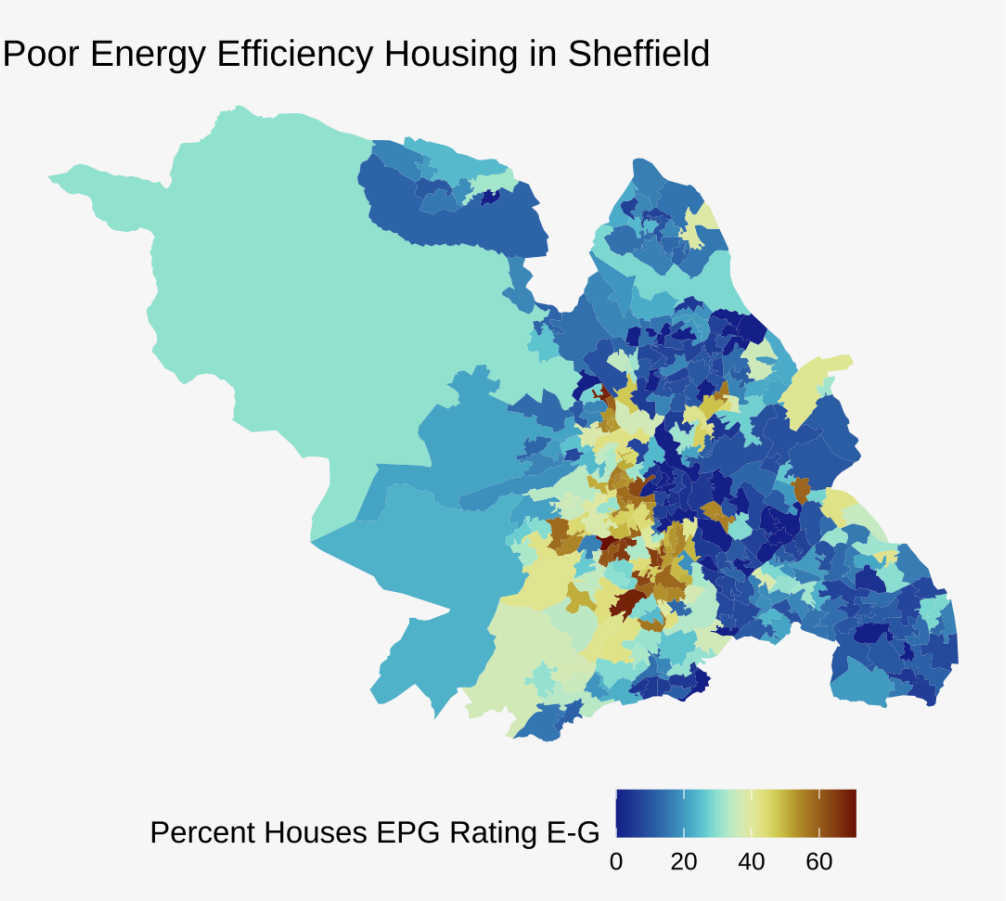
- Often an engaging way to present data — few people are interested in general patterns, everyone's interested in where they live.
- Can show interesting things in its own right — concentration or dispersion; social frontiers (Dean, et al. 2018)



What is spatial analysis and why learn it?

Argument 2: Because it's useful.

- Knowing the spatial characteristics of a social problem (how concentrated or dispersed it is) can help in designing effective policy. 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.

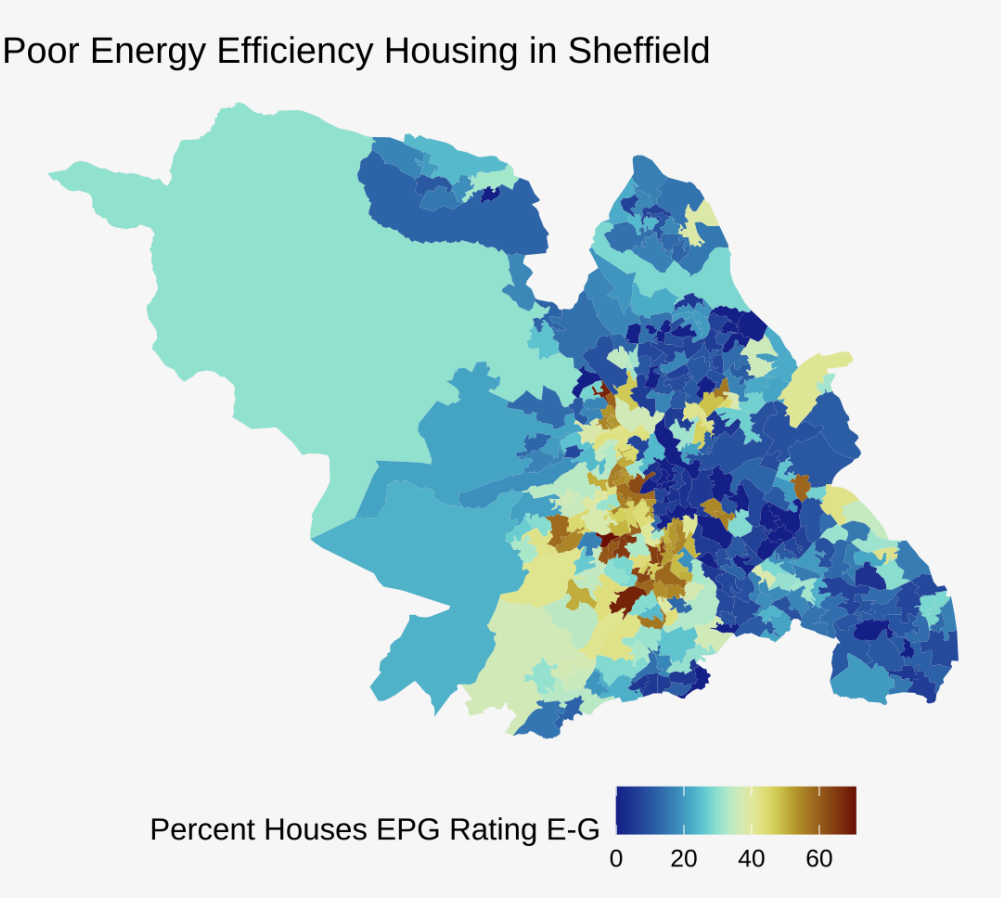


What is spatial analysis and why learn it?

Argument 3: Because it can cause some errors in our modelling

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

- Recall from Week 9 the concept of "autocorrelation" — in a regression model we assume that observations are independent of each other.
- However, if observations are spatially distributed, they will be related to eachother through proximity/distance (a Gaussian Process).



Week 11: Spatial Analysis — Part II

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

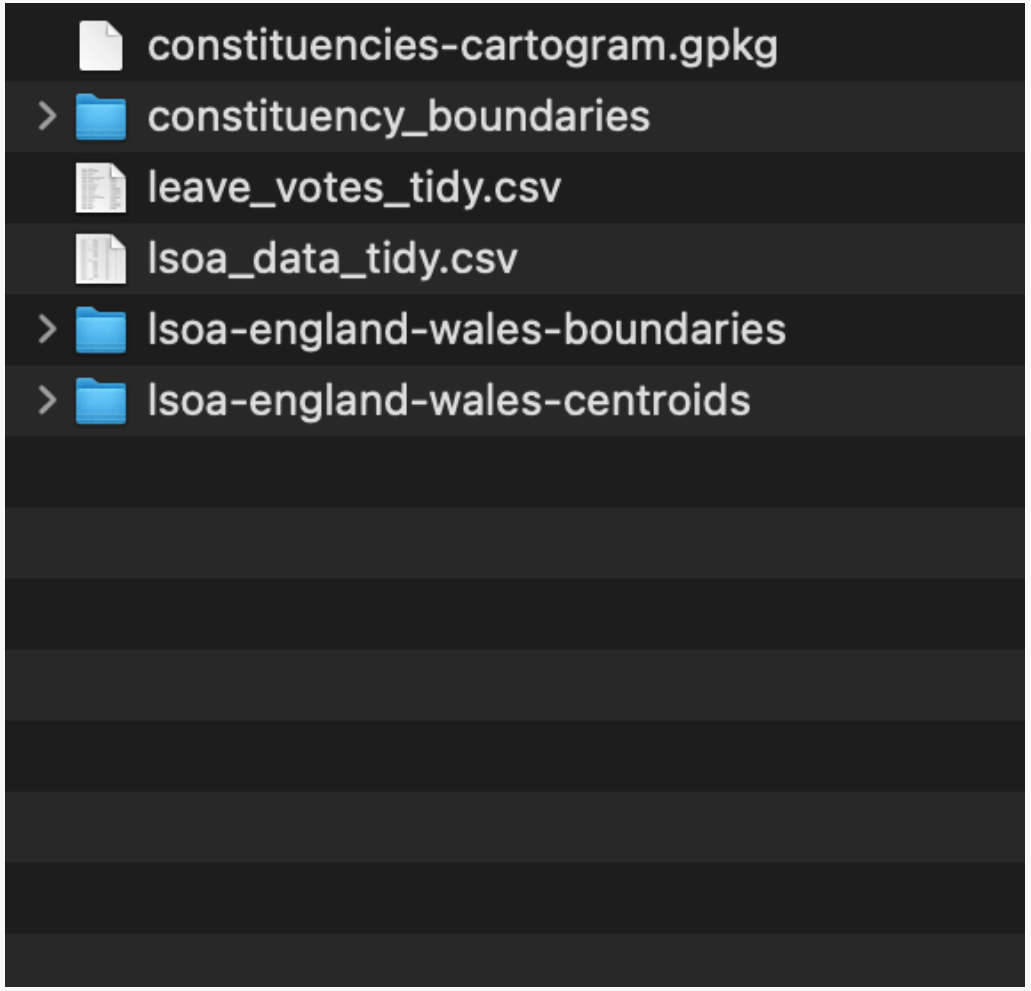

sf:
SPATIAL DATA ... SIMPLIFIED.
ATTRIBUTES + GEOMETRIES
1 Y 2.6 blue
6 N 5.0 red
12 Y 11.9 green
Y 13.8 blue
N 21.7 red
GLUE
tape
Sticky geometries: for people who love their maps and sanity.
Horst '18

Spatial data is... complicated.

- 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

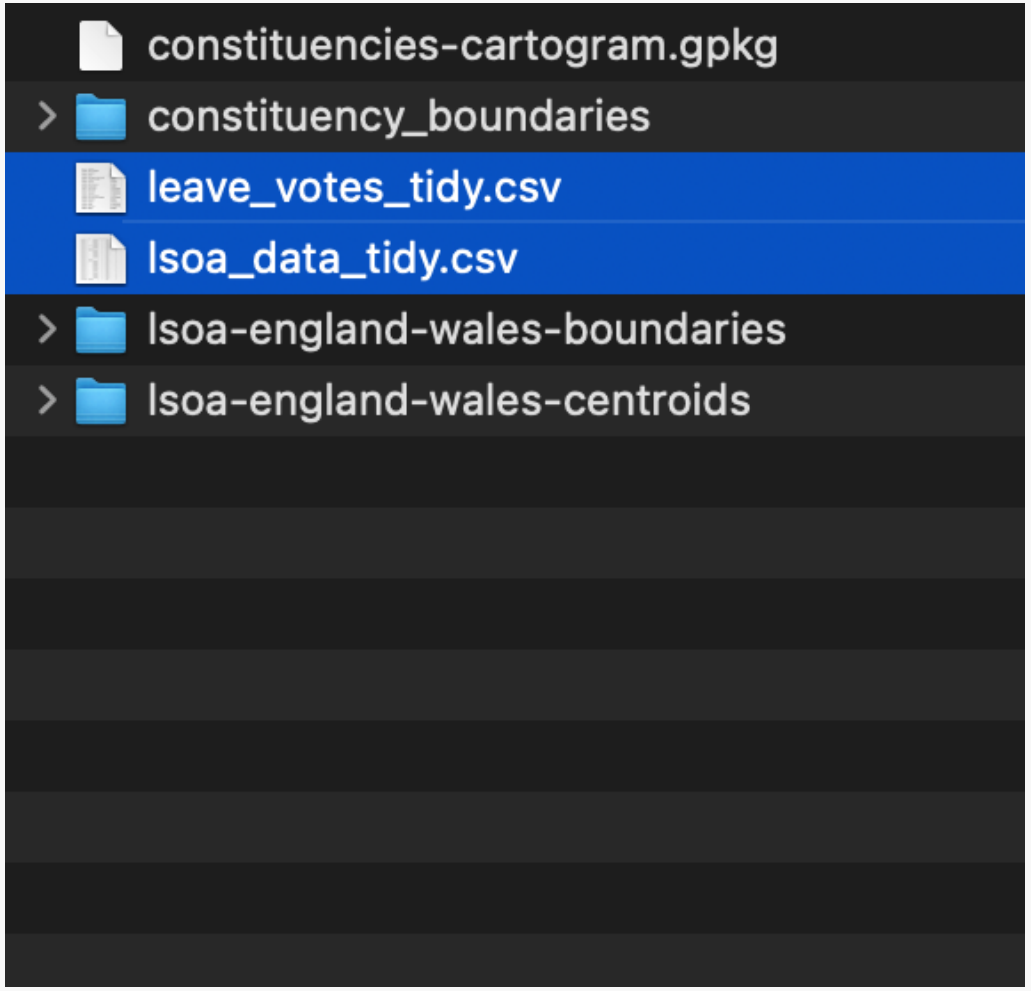
On the right is my data folder. Inside it are multiple different forms of spatial and non-spatial data.



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.

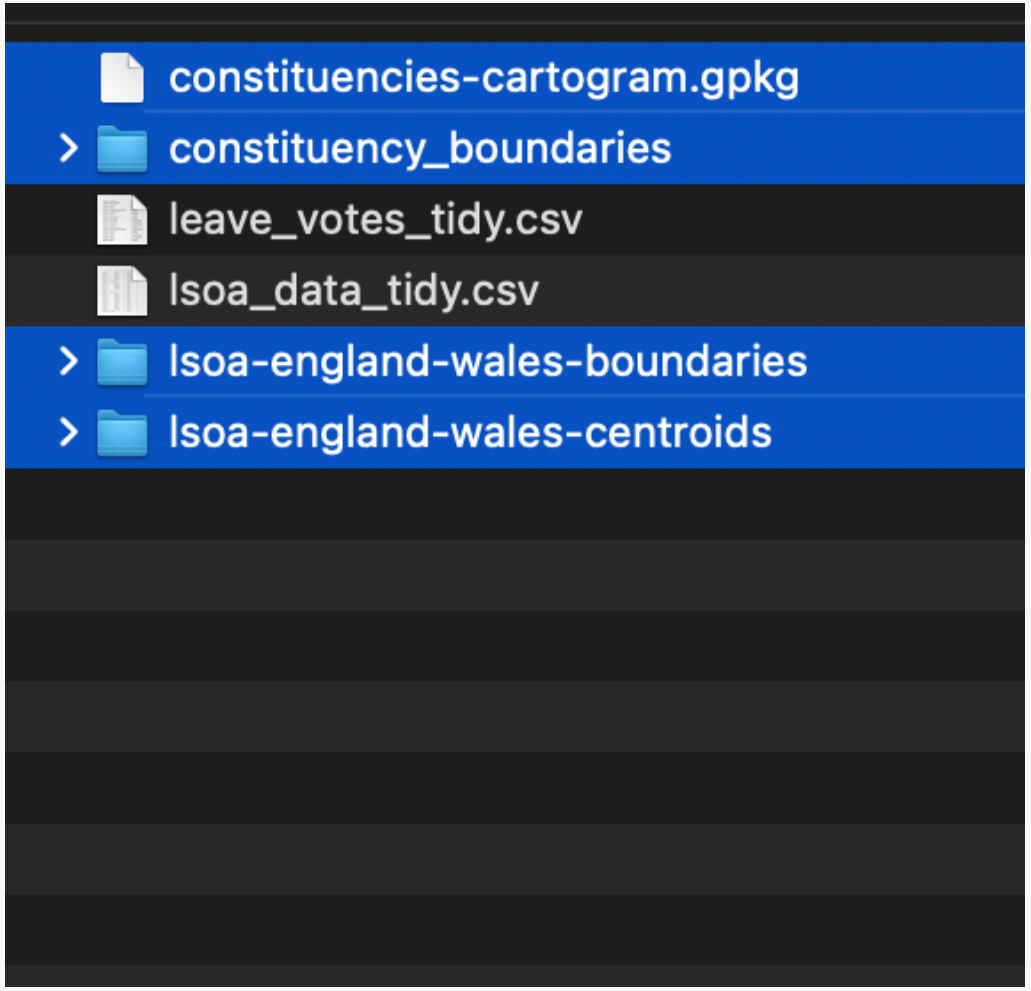
- These two files are just plain CSVs (comma-separated values). They contain my non-spatial data.
- These might be things I want to project onto the spatial boundaries (e.g. EU leave vote share by constituency; percentage of houses with E to G energy efficiency)
- Inportantly, they contain a **unique 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                      crs_name
## 1    1 Group names         Point       59      4 OSGB36 / British National Grid
## 2 2 Group outlines Multi Polygon       60      3 OSGB36 / British National Grid
## 3  3 City outlines Multi Polygon       55      4 OSGB36 / British National Grid
## 4 4 Constituencies Multi Polygon      650      6 OSGB36 / British National Grid
## 5     5 Background Multi Polygon        3      2 OSGB36 / British National Grid
## 6     layer_styles            NA        5     12                         <NA>
```

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                      crs_name
## 1    1 Group names         Point       59      4 OSGB36 / British National Grid
## 2 2 Group outlines Multi Polygon       60      3 OSGB36 / British National Grid
## 3  3 City outlines Multi Polygon       55      4 OSGB36 / British National Grid
## 4 4 Constituencies Multi Polygon      650      6 OSGB36 / British National Grid
## 5     5 Background Multi Polygon        3      2 OSGB36 / British National Grid
## 6     layer_styles            NA        5     12                         <NA>
```

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~CloudDocs/r/teaching-introduction-
##   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 ymax: 70.35025
## Projected CRS: OSGB36 / British National Grid
```

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

```
## Reading layer `4 Constituencies' from data source
##   `/Users/calumwebb/Library/Mobile Documents/com~apple~CloudDocs/r/teaching-introduction-
##   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 ymax: 69.48179
## 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_Generalised_Clipped__Boundari
##   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 ymax: 55.81107
## Geodetic CRS:  WGS 84
```

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

```
## Reading layer `Lower_Layer_Super_Output_Areas_(December_2011)_Population_Weighted_Centroi
##   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 ymax: 654394.9
## Projected CRS: OSGB36 / British National Grid
```

Week 11: Spatial Analysis — Part III

Spatial Analysis in R: Joining social science data to spatial data

Joining social science 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 social science 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 social science 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>
```

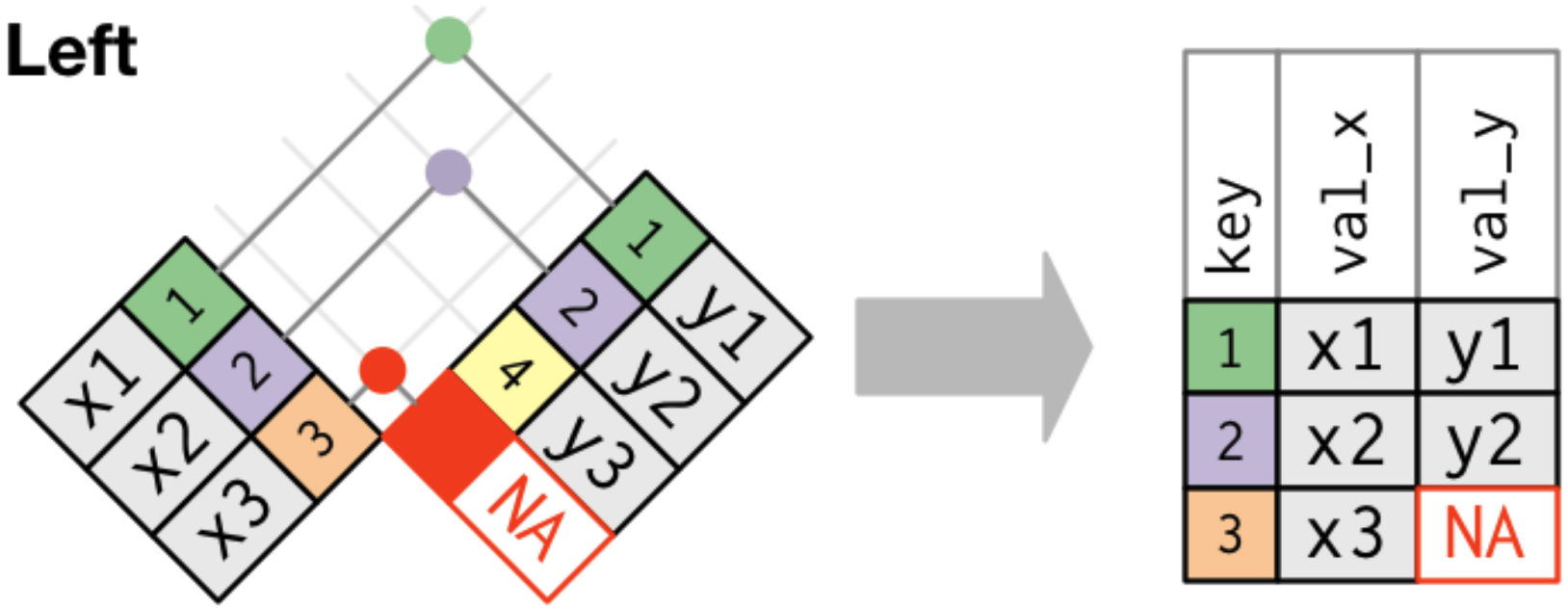
```
#
lsoa_boundaries
```

```
## 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 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 [°]>
```

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

Joining social science data to spatial data

`left_join` (Wickham & Grolemund, 2017)



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Joining social science 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

Draw the table below and fill in the gaps based on the rules of a `left_join`.

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

ID	age	smoker

Joining social science 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

Draw the table below and fill in the gaps based on the rules of a `left_join`.

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

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

Joining social science 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_code"))
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 11: Spatial Analysis — Part IV

Spatial Analysis in R: Spatial Data Visualisation — Choropleth Maps

Choropleth Maps

Okay, now we can finally start making maps!

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
```

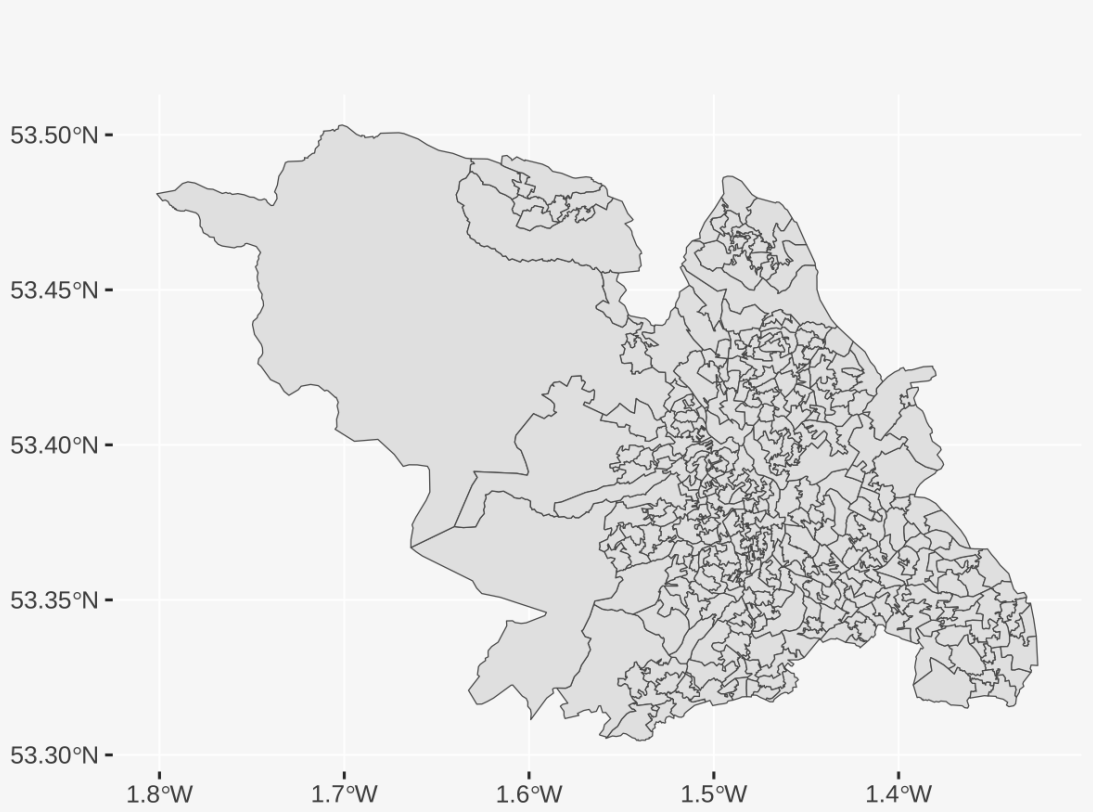
```
## Simple feature collection with 345 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1.801471 ymin: 53.30457 xmax: -1.324727 ymax: 53.50314
## Geodetic CRS:  WGS 84
## # A tibble: 345 × 16
##    objectid lsoa11cd  lsoa11nm       lsoa11nmw      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>,
## #   idaopi_2019 <dbl>, percent_eg_rated <dbl>, housesales_1998 <dbl>,
## #   housesales_2018 <dbl>, utla17nm <chr>, housesales_1998_quantiles <int>,
## #   housesales_2018_quantiles <int>, population <dbl>
```

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()
```

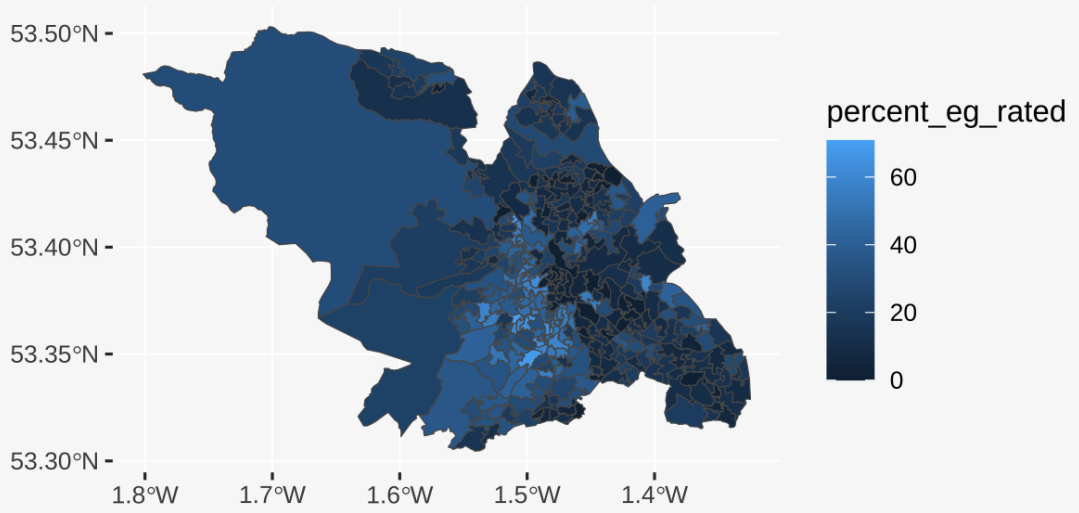


Choropleth Maps

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

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

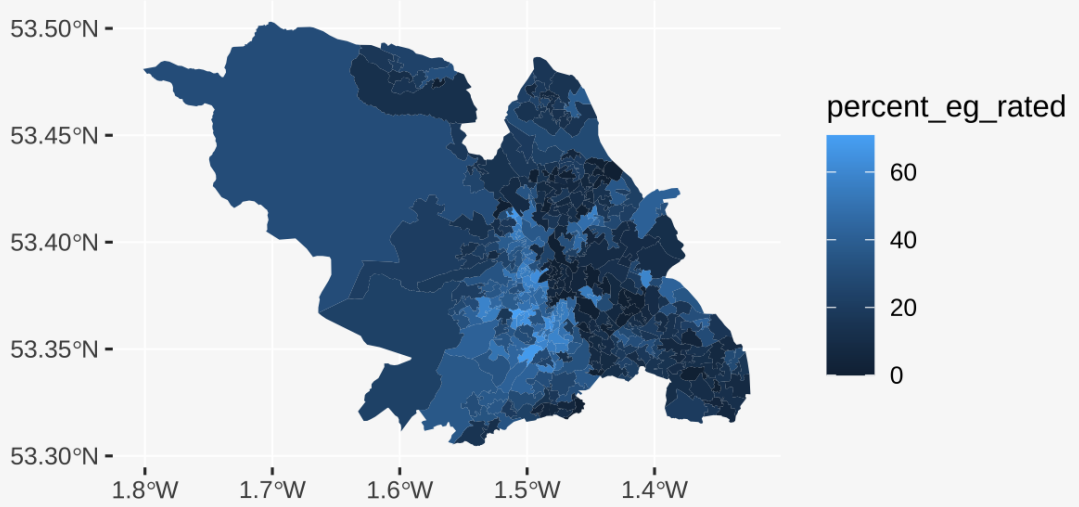
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated))
```



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"`.

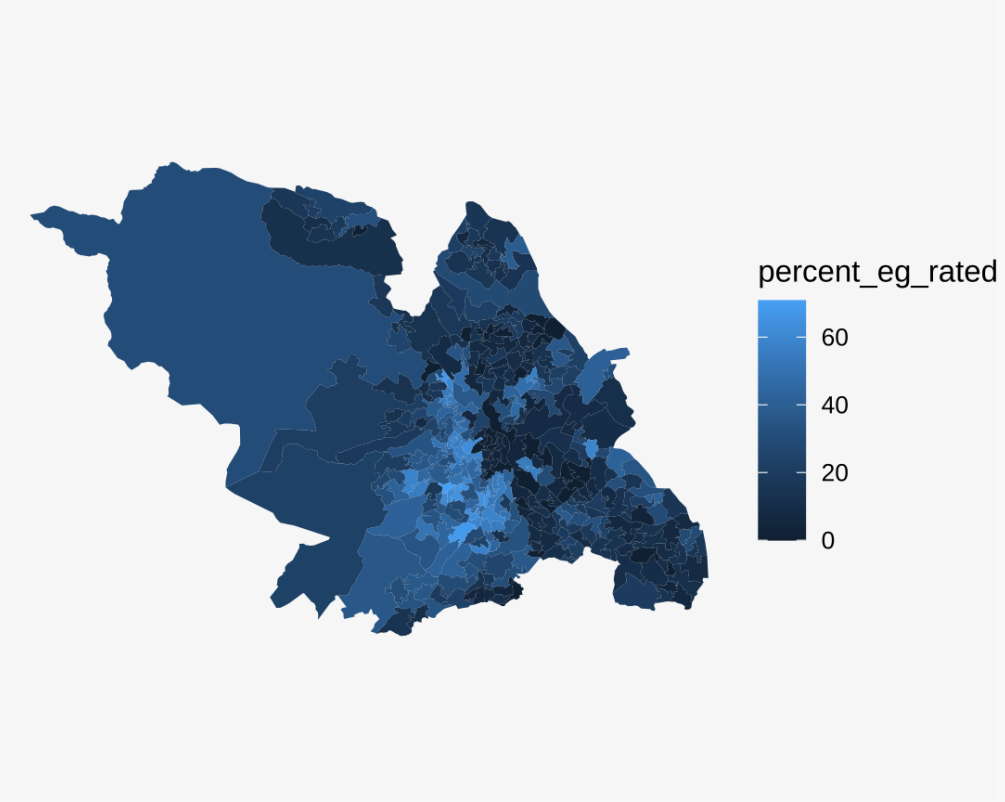
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent")
```



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.

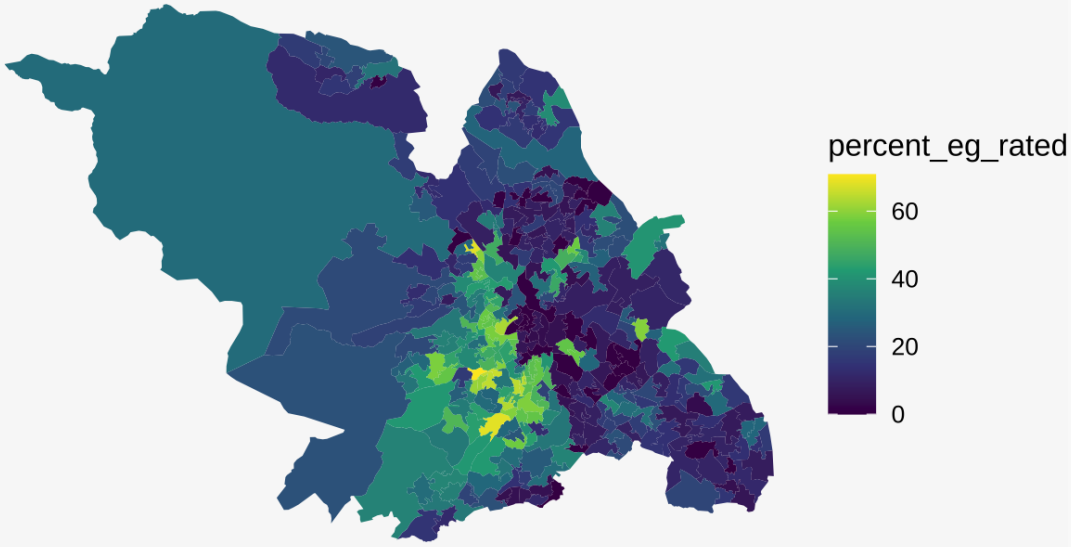
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent") +
  theme_void()
```



Choropleth Maps

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

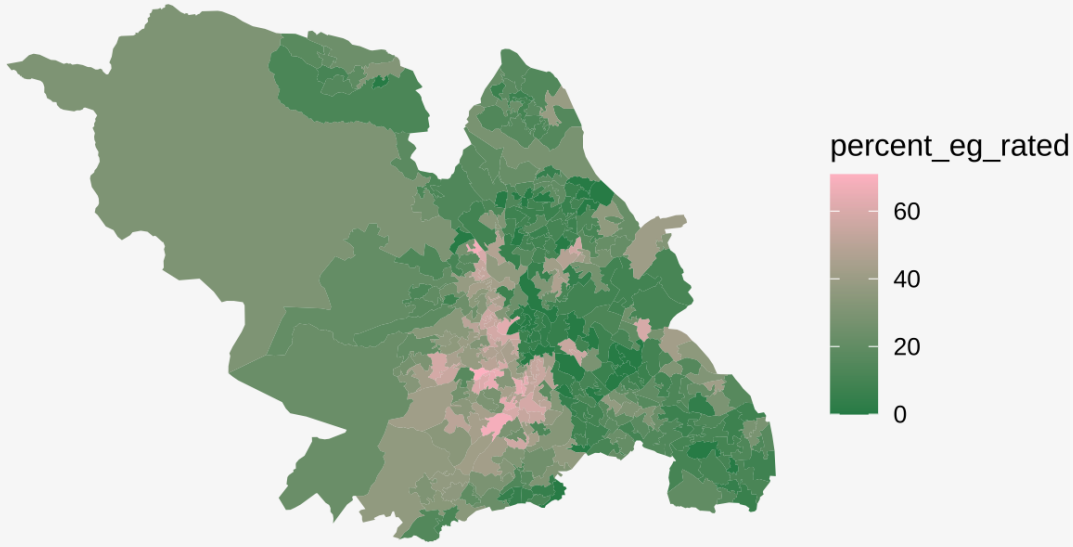
```
ggplot(data = sheffield_data) +
  geom_sf(aes(fill = percent_eg_rated), colour = "transparent") +
  theme_void() +
  scale_fill_viridis_c()
```



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 11: Spatial Analysis — Part V

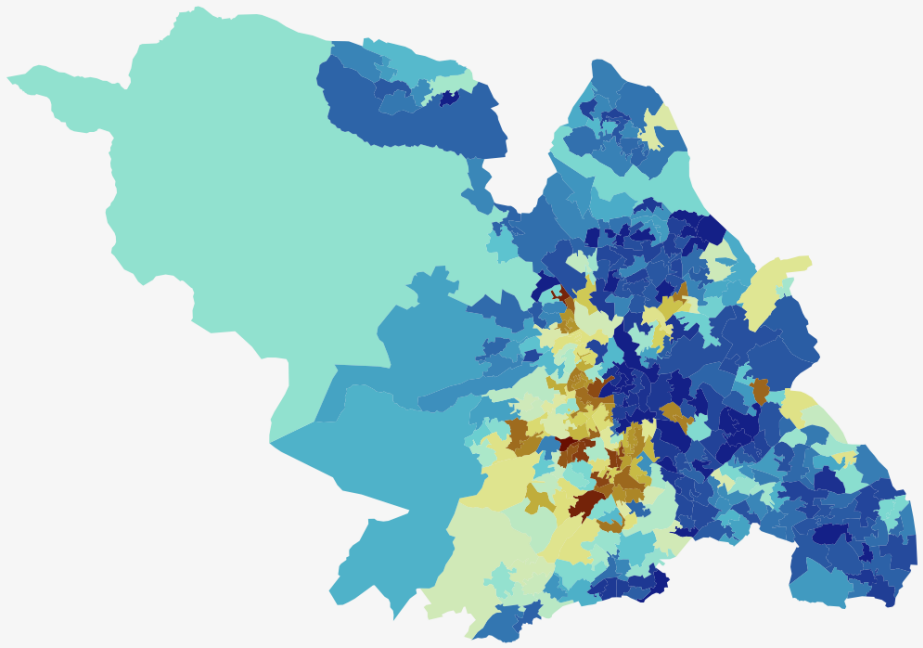
Spatial Analysis in R: Spatial Data Visualisation — Cartograms

Cartograms

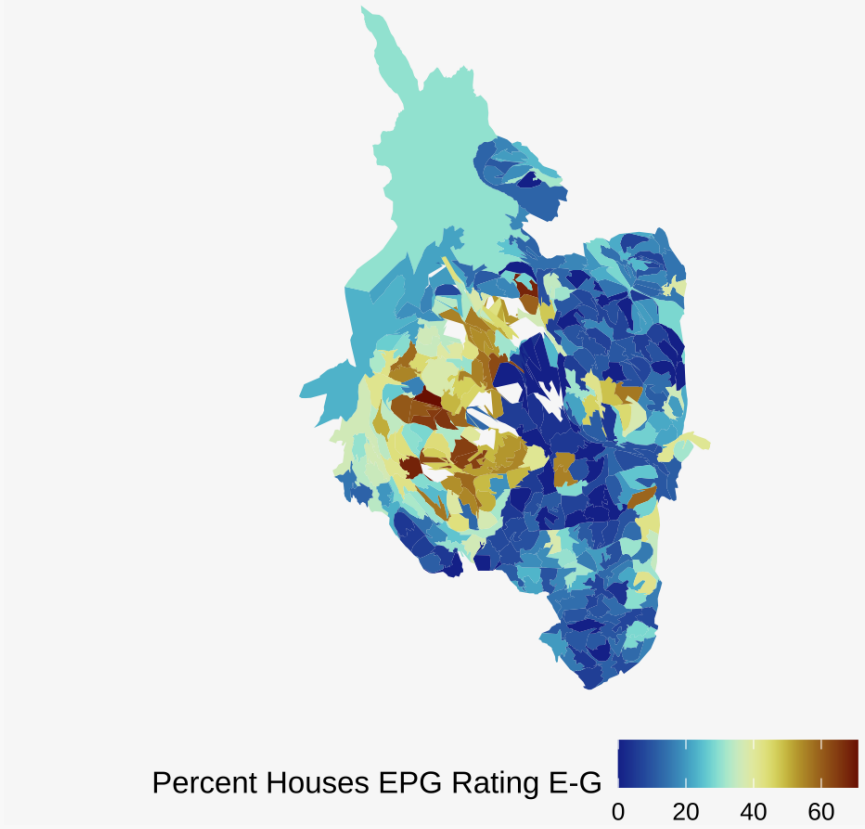
While a **choropleth map** refers to the colour-coding of a **geographically accurate** map, a **cartogram** refers to the colour-coding of a map that has been **transformed** in some way while (mostly) maintaining spatial arrangement, for example, to account for population size.

Cartograms

Choropleth

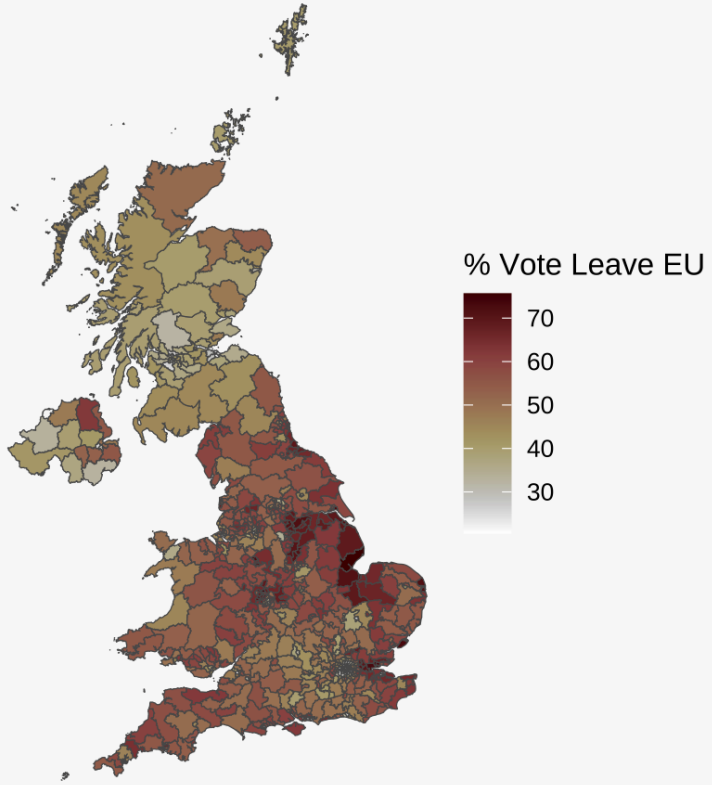


'Rubber Sheet' Population Cartogram

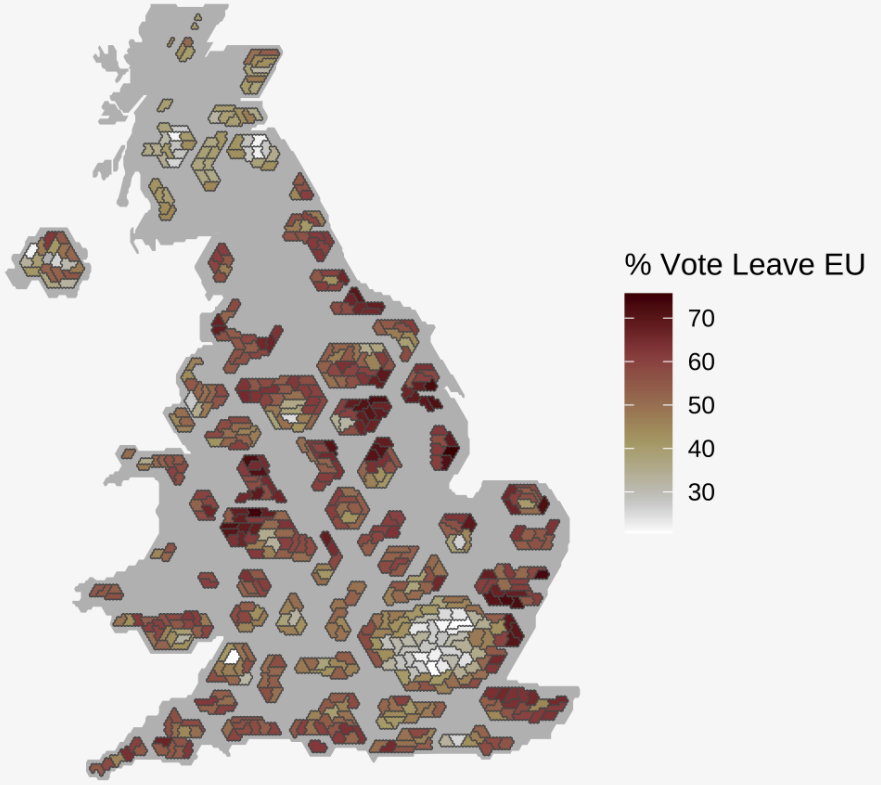


Cartograms

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 11: Spatial Analysis — Part V

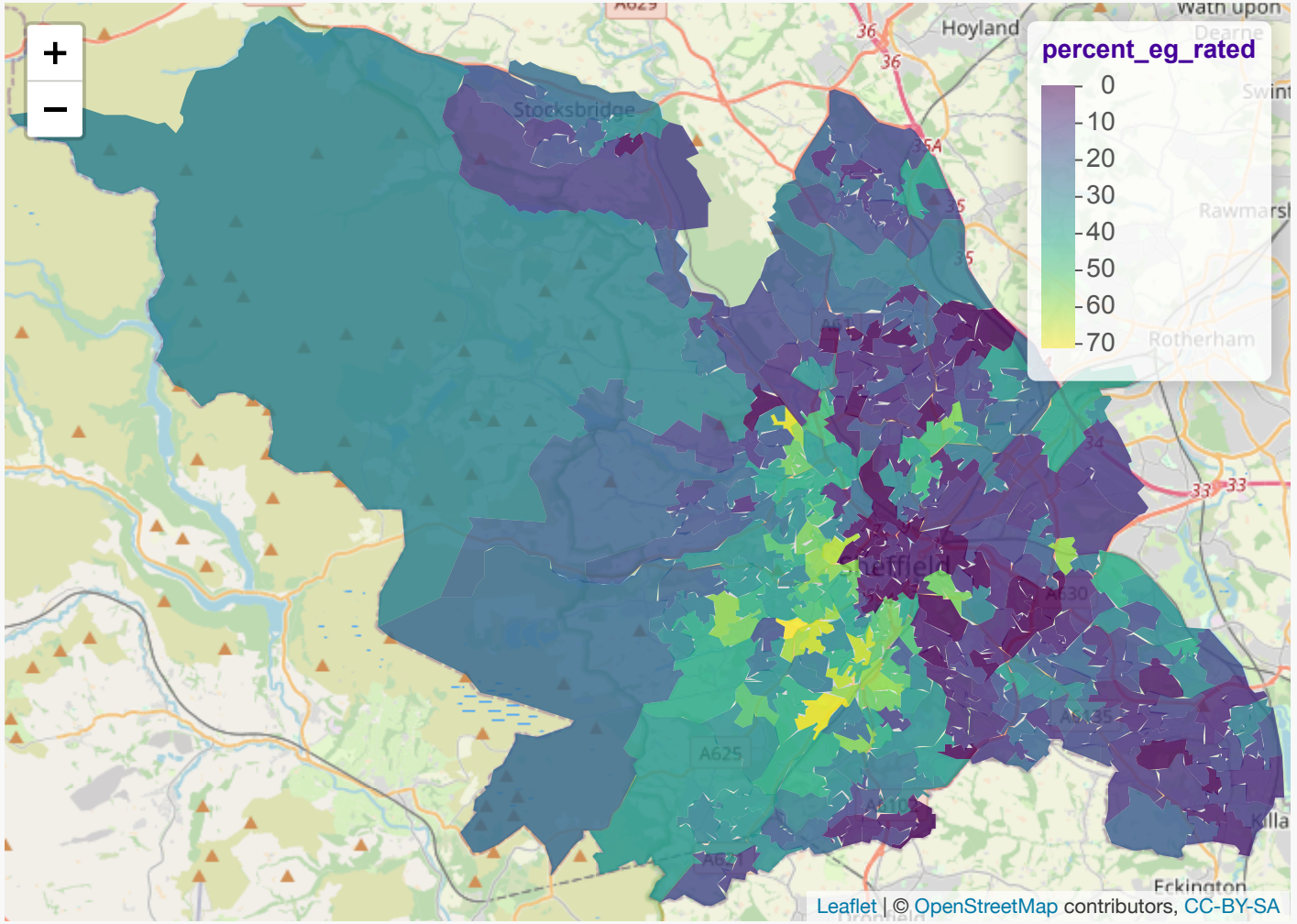
Spatial Analysis in R: Spatial Data Visualisation — Interactive

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.



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)
```

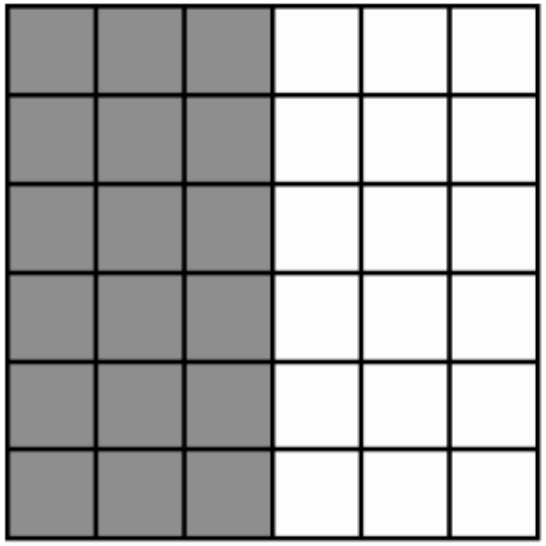
Week 11: Spatial Analysis — Part VI

Spatial Analysis in R: Moran's I

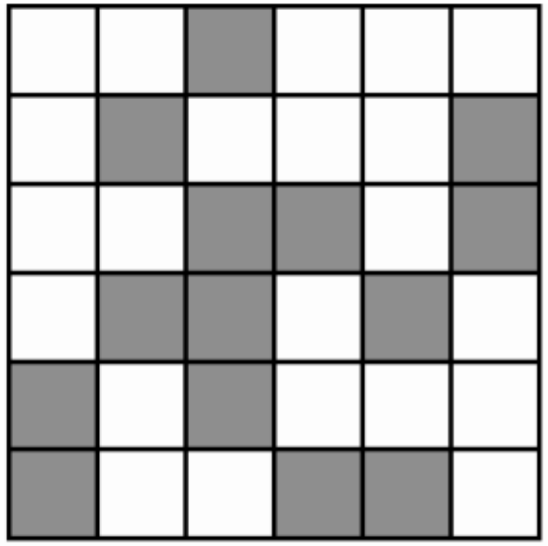
But how can I express the spatial relationship of a variable?

Moran's I

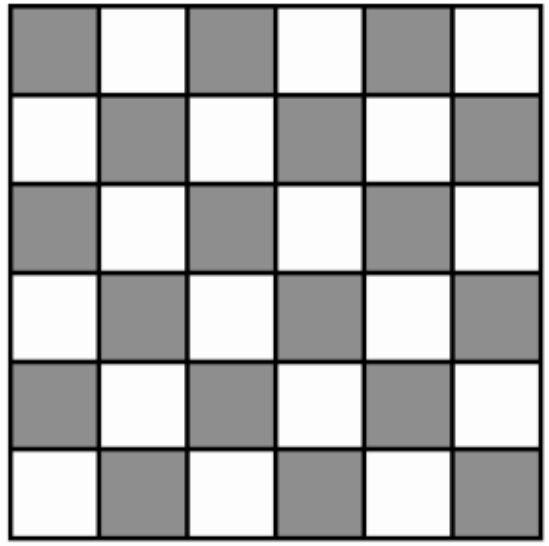
Moran's I is a statistical measure of the autocorrelation in spatial data.



Positive spatial autocorrelation



No spatial autocorrelation



Negative spatial autocorrelation

Source: Kirkegaard, 2015

Moran's I

Data Preparation

To calculate Moran's I, we need three things:

- The **variable** we're interested in finding the spatial autocorrelation of.
- The population-weighted or geometric **centroids** of our boundaries (or a list of their bordering neighbours, but we are going to use the centroid approach).
- A closeness matrix (the inverse of the distance matrix we used last week in factor analysis) which tells us the **closeness of each centroid to every other centroid**

Moran's I

Data Preparation

I'll start by reading in the centroid data and joining it to the LSOA data, using what we covered in the second and third part of this lecture. I'll also filter the data to be just Sheffield again.

I'm also going to re-calculate the house sales quantiles (as they are currently national popularity) — recalculating them on the Sheffield only data will give me house buying popularity only relative to areas in Sheffield.

```
sheff_cent <- read_sf("data/data_dummy/lsoa-england-wales-centroids/") %>%
  filter(str_detect(lsoa11nm, "Sheffield"))

sheff_cent <- left_join(sheff_cent, lsoa_data,
          by = c("lsoa11cd" = "lsoa_code")) %>%
  mutate(housesales_1998_quantiles = ntile(housesales_1998, 20),
         housesales_2018_quantiles = ntile(housesales_2018, 20))

sheff_cent
```

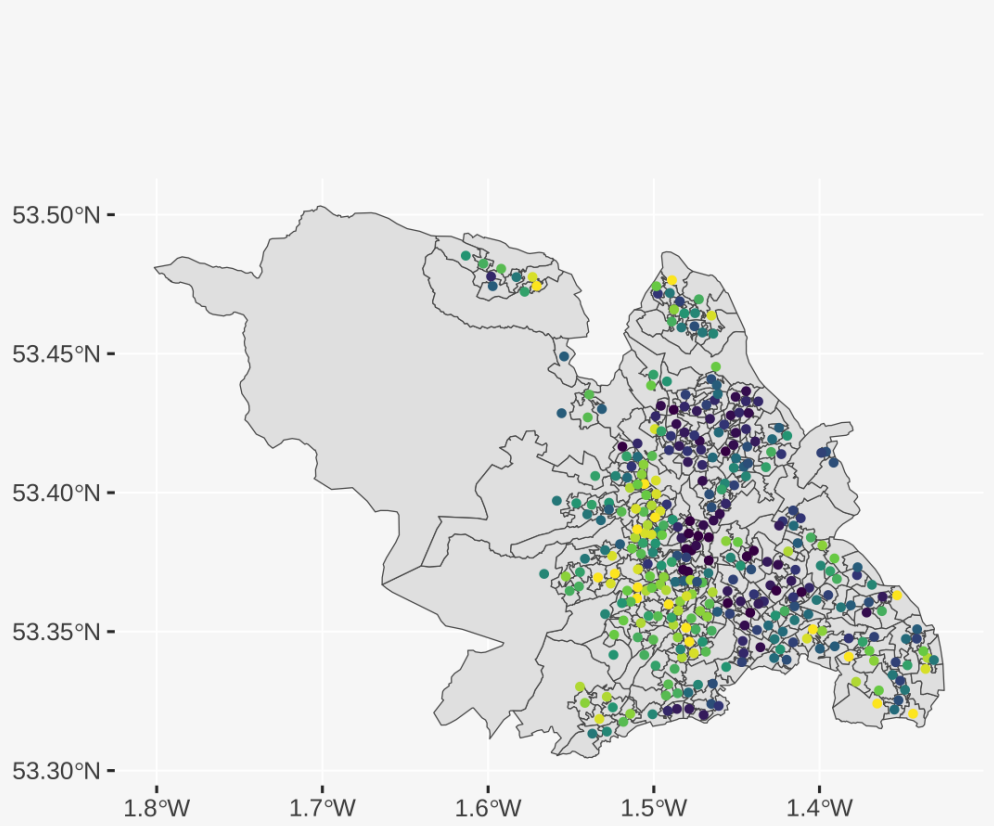
```
## Simple feature collection with 345 features and 12 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 425746.8 ymin: 379695.1 xmax: 444644.8 ymax: 398795.3
## Projected CRS: OSGB36 / British National Grid
## # A tibble: 345 × 13
##    objectid lsoa11cd  lsoa11nm              geometry lsoa_name      idaopi_2019
##  *    <int> <chr>     <chr>              <POINT [m]> <chr>                <dbl>
##  1     6155 E01008066 Sheffield… (433535.7 387806.6) Sheffield 028B        20.4
##  2     6156 E01008067 Sheffield… (433683.6 387664.8) Sheffield 030D        27.7
##  3     6157 E01008064 Sheffield…   (432992.2 387736) Sheffield 028A        12
##  4     6158 E01008062 Sheffield… (436615.4 391780.3) Sheffield 013E        34
##  5     6159 E01008063 Sheffield…   (435927.4 391785) Sheffield 011B        11.2
##  6     6160 E01008060 Sheffield… (436404.2 392473.2) Sheffield 010D        30.5
##  7     6161 E01008061 Sheffield… (436145.7 392102.4) Sheffield 011A        26.5
##  8     6162 E01008068 Sheffield… (434465.2 387552.6) Sheffield 026B        40.8
##  9     6163 E01008069 Sheffield…   (433265.3 387676) Sheffield 028C        13.7
## 10     6165 E01032585 Sheffield… (443074.4 380751.8) Sheffield 072E         7.2
## # ℹ 335 more rows
## # ℹ 7 more variables: percent_eg_rated <dbl>, housesales_1998 <dbl>,
## #   housesales_2018 <dbl>, utla17nm <chr>, housesales_1998_quantiles <int>,
## #   housesales_2018_quantiles <int>, population <dbl>
```

Moran's I

Data Preparation

I'll start by reading in the centroid data and joining it to the LSOA data, using what we covered in the second and third part of this lecture. I'll also filter the data to be just Sheffield again.

I'm also going to re-calculate the house sales quantiles (as they are currently national popularity) — recalculating them on the Sheffield only data will give me house buying popularity only relative to areas in Sheffield.



Moran's I

Data Preparation

Next, I need to get our X and Y coordinates from our data. I can use the `st_coordinates()` function to do this easily.

```
# Get X and Y coordinates
sheff_cent_coords <- st_coordinates(sheff_cent)

head(sheff_cent_coords)
```

```
##             X        Y
## [1,] 433535.7 387806.6
## [2,] 433683.6 387664.8
## [3,] 432992.2 387736.0
## [4,] 436615.4 391780.3
## [5,] 435927.4 391785.0
## [6,] 436404.2 392473.2
```

Moran's I

Data Preparation

Next, I need to calculate our Euclidean ('as the crow flies') distance matrix. I can use the `dist` function built into `R` to do this. This is easier than using the `daisy` function from last week as we need to do further manipulation to make it a **closeness** matrix. I also need to change it to explicitly be a `matrix` class object for this manipulation.

```
# Calculate Euclidean distance matrix
sheff_dist <- dist(sheff_cent_coords)

# Change to be matrix object (for further manipulation)
sheff_dist <- as.matrix(sheff_dist)
```

Moran's I

Data Preparation

Now I need to **invert the distance matrix**, so that it becomes a "closeness" matrix. This is easy to do with the following two steps:

- Calculate 1 divided by every value in the distance matrix
- Replace the diagonal values in the new matrix with 0

```
# Calculate inverse of distance
sheff_close <- 1/sheff_dist
# Set diagonals to 0
diag(sheff_close) <- 0

# Preview output
sheff_close[1:5, 1:5]
```

```
##              1            2            3            4            5
## 1 0.0000000000 0.0048817702 0.0018247218 0.0001989083 0.0002154238
## 2 0.0048817702 0.0000000000 0.0014387929 0.0001979033 0.0002131501
## 3 0.0018247218 0.0014387929 0.0000000000 0.0001841664 0.0001999619
## 4 0.0001989083 0.0001979033 0.0001841664 0.0000000000 0.0014535795
## 5 0.0002154238 0.0002131501 0.0001999619 0.0014535795 0.0000000000
```

Moran's I

Data Preparation

We can finally calculate the Moran's I statistic using the `spdep` (Spatial Dependence) package.

The `moran.test` function needs two arguments:

- `x` = The variable of interest.
- `listw` = The closeness matrix. Note that we have to put our matrix in the `mat2listw()` function from `spdep` to turn it into the necessary kind of class for it work.

```
library(spdep)

# Disable scientific notation for less than 10 length
options(scipen = 10)

# Moran's I statistic for Housing Sales 1998
moran.test(x = sheff_cent$housesales_1998_quantiles,
           listw = mat2listw(sheff_close))
```

```
##
##     Moran I test under randomisation
##
## data:  sheff_cent$housesales_1998_quantiles
## weights: mat2listw(sheff_close)
##
## Moran I statistic standard deviate = 29.866, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##     0.12980672548    -0.00290697674     0.00001974588
```

Moran's I

Morans I = **1**, Highly Clustered Spatially; Moran's I = **-1**, Highly Dispersed; Moran's I = **0**, Random Distribution in Space

```
# Moran's I statistic for Housing Sales 1998
moran.test(x = sheff_cent$housesales_1998_quantiles,
           listw = mat2listw(sheff_close))
```

```
##
##     Moran I test under randomisation
##
## data:  sheff_cent$housesales_1998_quantiles
## weights: mat2listw(sheff_close)
##
## Moran I statistic standard deviate = 29.866, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##     0.12980672548    -0.00290697674     0.00001974588
```

```
# Moran's I statistic for Housing Sales 2018
moran.test(x = sheff_cent$housesales_2018_quantiles,
           listw = mat2listw(sheff_close))
```

```
##
##     Moran I test under randomisation
##
## data:  sheff_cent$housesales_2018_quantiles
## weights: mat2listw(sheff_close)
##
## Moran I statistic standard deviate = 15.495, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##     0.06594730658    -0.00290697674     0.00001974588
```

Have the areas where people buy the most property in Sheffield grown more or less spatially concentrated over time?

Week 11: Spatial Analysis — Part VII

Spatial Analysis in R: Taking Spatial Analysis Further with Spatial Regression Models

Spatial Regression Models

The effect of spatial autocorrelation can be controlled for by means of a spatial regression model. This, however, is a quite advanced method.

```
simple_model <- lm(data = sheff_cent,
   formula = percent_eg_rated ~ idaopi_2019)

summary(simple_model)
```

```
##
## Call:
## lm(formula = percent_eg_rated ~ idaopi_2019, data = sheff_cent)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.679 -12.944  -4.964  11.240  45.635
##
## Coefficients:
##             Estimate Std. Error t value        Pr(>|t|)
## (Intercept) 32.49250    1.60226  20.279         < 2e-16 ***
## idaopi_2019 -0.43183    0.06452  -6.693 0.0000000000891 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.98 on 343 degrees of freedom
## Multiple R-squared:  0.1155,    Adjusted R-squared:  0.1129
## F-statistic:  44.8 on 1 and 343 DF,  p-value: 0.00000000008915
```

```
library(spaMM)
sheff_cent_sr <- sheff_cent
sheff_cent_sr$x <- sheff_cent_coords[,1]
sheff_cent_sr$y <- sheff_cent_coords[,2]
m_spamm <- fitme(percent_eg_rated ~ idaopi_2019 +
                   Matern(1 | x + y),
                 data= sheff_cent_sr, family= "gaussian")
summary(m_spamm, verbose = FALSE)$beta_table
anova(m_spamm)
```

```
##               Estimate   Cond. SE   t-value
## (Intercept) 27.4432820 2.34413226 11.707224
## idaopi_2019 -0.3798998 0.06023976 -6.306463
```

```
## Result accounting for uncertainty for the following estimates of nuisance parameters:
##  corrPars.1.nu corrPars.1.rho       lambda.1            phi
##    0.493907678    0.001242692  231.594233350    0.002549916
```

```
## Type II Analysis of Variance Table with Satterthwaite's method
##               Sum Sq Mean Sq NumDF  DenDF F value         Pr(>F)
## idaopi_2019 0.10141 0.10141     1 247.65  39.772 0.000000001302 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Week 11: Spatial Analysis — Part VIII

Summary & Practical

Summary

- Spatial analysis can include an **interesting new dimension in your research project** — either through visualising your findings to make them easier to communicate; using it to complement qualitative research designs; or for incorporating spatial autocorrelation to re-visit classic research studies.
- 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 population size.
- We can summarise **spatial autocorrelation** using the **Moran's I** statistic, where -1 indicates total dispersion and +1 indicates total concentration of a variable.
- Spatial autocorrelation and analysis can be taken further using **spatial regression models**, which can account for the impact of spatial dependence. Though they are too advanced to cover in this module, they are not too difficult to get working in **R**.

Practical

In this week's practical, we are going to be recreating the maps of Sheffield areas based on house buying popularity, but you will also be following along and adapting the code to replicate this analysis for a city or region of your choice.

- Download and unzip the `week-11-exercise.zip` file from the Blackboard page. Open the `week-11-exercise.Rproj` Rproject file and open the accompanying .Rmd file.

Thank you and well done!

A reminder of where we started...

Be like Florence.

"Whenever I am infuriated, I revenge myself with a new diagram."

"It's worth noting, before getting started, that this material is hard. If you find yourself confused at any point, you are normal. Any sense of confusion you feel is just your brain correctly calibrating to the subject matter. Over time, confusion is replaced by comprehension..."

Richard McElreath. (2020). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan. 2nd Ed.*. Boca Raton, FL: CRC Press. p.193

Additional Resources & Workshops from the University

Library

Research Skills and Critical Thinking (RSCT) workshops

- Discovering Information: A masterclass
- Referencing and preventing plagiarism
- Using reference management software: an introduction

Online resources

- Developing a search strategy
- Producing a literature review
- Tutorials A-Z

301 Skills Centre

- Essay structure & planning
- Developing an academic argument
- Reflecting on your academic progress
- Making the most of feedback

M.A.S.H.

- Statistics resources:
- 1-to-1 Support
- General statistics workshops

Career's Service

- Reflecting & recording personal development