Data Visualisation Week 7: Making maps

What's happening today?

Maps! Today we're making maps.

We're also starting to load data *locally*, rather than from the web. In the first half of the semester, we pulled data from the web; this is fine when we've got a web host, or what we need is just a small file, but when we're working with more or bigger files this often isn't practical.

In addition to this, we're going to combine a couple of datasets together. We'll go into some more detail in the coming weeks about how to do this.

This file accompanies a .zip file containing a few different things. We can start with a file called "code for week 7.R", which we'll run. The main thing to know here is that you're using a .Rproj file, which means that a few different files are fitting together. It's one way around the often fiddly problem of *setting the working directory*, which is one approach to telling R where to look for files.

You will need to **unzip** the .zip file before we can start. If you're on an Apple machine, this is easy - you can just double click on the file and it'll unzip into a new folder. If you're on a Windows machine, you'll need to right-click and then click "Extract All". If you just double click on the zip file itself, it won't work.

Health warning

Broadly, drawing maps is more computationally intensive than drawing the other kinds of graphs that we're drawing on this module. For this reason, it generally takes longer for ggplot2 to draw the maps we're doing this week than it has done for the graphs we did in the first part of the semester, and sometimes won't draw them at all. This is particularly the case if you're on a Mac rather than on a Windows machine. If this happens, don't panic; it's normal.

Getting started

Let's get started by loading some packages and data.

```
library(tidyverse)
library(sf)
```

The **sf** package is new. **sf** standards for **simple features**, and it's a package specifically for geographic data using R. (There's lots of different software packages for the visualisation of maps in particular – ArcGIS is the market leader, while I'm keener on QGIS, which, like R, is free and open source.) By contrast, you've seen tidyverse before. (As is often the case, you might have to install sf.)

Let's now load some data.

```
westminster <- read_sf(dsn = ".", layer = "westminster_const_region")
airports <- read_sf(dsn = ".", layer = "airports_gb")

housing <- read.csv("housing.csv")

h_and_w <- inner_join(westminster, housing, by = "CODE")
```

OK, there's a few things going on here.

Loading **housing** is relatively straightforward; it's a read.csv() command, with a csv name in the folder. You'll note that it's not a URL, like you've seen before; that's because it's in the same folder as this R project.

Loading **westminster** is a bit more complicated. You'll note that there's several different files in this folder that start "westminster_const_region"; this is because *shapefiles*, which are the flavours of spatial data that we're working with, tend to consist of several different files, including geography, projections, and databases with data frames in. So there's two elements in the bracket. The first, **dsn**, specifies where we're looking; the . means we're looking in the same folder as the files we're using. (I'd generally recommend this.) The second, **layer**, asks for the name of the layer we're working with; in this instance, that's westminster_const_region.

Loading **airports** is exactly the same as loading **westminster**, as it's another shapefile.

Finally, we loaded **housing**, which is just a csv like the ones you've seen before.

Merging

The last thing we did was to *merge* two files. **westminster** is a shapefile of parliamentary constituencies, without much information about them; **housing** is a data frame that contains information about how many people in each constituency owned their home at the time of the 2011 Census, without much information about what those constituencies look like. In *merging* the two files, we've now got a shapefile which includes information about how many people own their homes. (We'll do this more systematically next week.)

Drawing maps

Let's draw some maps. Specifically, let's start by drawing a map of where different airports are.

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

You'll note that this is the shortest ggplot we've run so far; there's no aesthetic mappings. What we have is a bunch of simple features; each observation in the airports data comes with longitude and latitude, which are effectively x and y coordinates.

While this probably looks vaguely like a map of the UK, it's not exactly what we're aiming for. (If you don't know much about the Scottish islands, the area north of Hadrian's Wall might be a bit confusing.)

What we can do as an alternative is draw a map of each of Britain's parliamentary constituencies, using the **westminster** data, like so. (This will probably take a little while to load; don't panic if that happens. If you're at home, this might be a good opportunity to make a cup of tea.)

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

OK, this is looking familiar! Let's colour it in.

Colouring in

How do we add an aesthetic mapping to a map? The same way we've been adding aesthetic mappings to everything else. Let's extend what we've done here by colouring in each of the UK's parliamentary constituencies according to the numbers of people who own their homes, as opposed to renting (or living rent free, etc.)

```
ggplot(data = h_and_w)+
  aes(fill = percent_owned) +
  geom_sf()  +
  scale_fill_viridis_c()
```

This is exactly the same kind of thing we've seen before. We've specified data, an aesthetic mapping (putting % owned in fill), a geometric object (sf), and tidied up a bit (changed the colour scheme to viridis).

Binning colour schemes

We're currently using a continuous colour scheme. If you've got incredibly sharp eyes, you can distinguish between an area that has 61% of its residents in owned homes and an area with 62%.

But most people can't. There's also a problem here in that there's a few areas (mostly in London) with very low rates of home ownership - around a third - that distort the overall picture.

As an alternative, let's bin these categories. There's two approaches to binning that I want to flag; there's others, but this is a starting point.

```
h_and_w %>%
  mutate(percent_owned_bin = cut_interval(percent_owned, 6)) %>%
  ggplot() +
  geom_sf(aes(fill = percent_owned_bin)) +
  scale_fill_viridis_d()
```

What have we done here?

You'll remember mutate() from before, as a method of generating new variables. What mutate's doing here is generating a new variable – percent_owned_bin – based on the percent_owned variable. Instead of doing maths to it, such as by dividing as we saw before, we're cutting it in such a way that we're generating a categorical variable, using the **cut_interval** function. Within this bracket, we're specifying which variable we want to turn into a discrete variable, and specifying the number of bins we want. Following this, we continue as before, running a ggplot with our new variable. The only difference is we're now using scale_fill_viridis_**d**, because this is now a discrete rather than continuous variable.

Here, you'll see we have six categories with equal ranges: 20% to 31%, 31% to 42%, 42% to 53%, and so on. But there's more constituencies in the higher categories than the lower ones; you'll see most of the map is green or yellow.

What if, instead, we wanted *quartiles*; what if we wanted to generate categories that would have equal numbers of constituencies in each? We can do this by changing **cut_interval** – which generates categories of equal intervals, but different numbers of observations – for **cut_number**, which generates categories of equal numbers of observations, but different intervals. Like so.

```
h_and_w %>%
  mutate(percent_owned_bin = cut_number(percent_owned, 6)) %>%
  ggplot() +
  geom_sf(aes(fill = percent_owned_bin)) +
  scale_fill_viridis_d()
```

(It's worth noting that there are alternative methods to coming up with class intervals than these two. **Jenks** is a popular one, otherwise known as **natural breaks**. If you're interested in this, it's worth looking into the **classInt** package.)

Last couple of things

First, let's tidy up that last graph. The intervals look OK, but they're a bit messy. We've seen the labs() command before, which can address the title of the legend; let's also address the labels for each of the categories.

```
h_and_w %>%
  mutate(percent_owned_bin = cut_number(percent_owned, 6)) %>%
  ggplot() +
  geom_sf(aes(fill = percent_owned_bin)) +
  scale_fill_viridis_d(labels =
                       c("less than 55%",
                         "55%-63%",
                         "63-67%",
                         "67-70%",
                         "70-73%",
                         "more than 73%")) +
  labs(fill = "% of households owned
(including with a mortgage)",
        title = "Lots of renters in urban areas")
```

This is starting to look a bit better. (There's still an issue with respect to the level of detail – it's easier to see what's going on in rural parts of Wales than it is in densely populated urban areas, for example – but we'll come back to that.)

One last thing is **coordinate systems**. We'll talk about this in next week's lecture, but by definition any map on a screen involves a certain level of distortion, as we're portraying a three-dimensional object in two-dimensional space. (This gets worse as the amount of the world we're drawing gets bigger; a map of Sheffield is probably less distorted than a map of the northern hemisphere.)

We can tweak coordinate systems in ggplot2 and in sf like so.

```
h_and_w %>%
  mutate(percent_owned_bin = cut_number(percent_owned, 6)) %>%
  ggplot() +
  geom_sf(aes(fill = percent_owned_bin)) +
  scale_fill_viridis_d(labels =
                         c("less than 55%",
                           "55%-63%",
                           "63-67%",
                           "67-70%",
                           "70-73%",
                           "more than 73%")) +
  labs(fill = "% of households owned
(including with a mortgage)",
       title = "Lots of renters in London") +
  coord_sf(crs = 3857)
```

What we've done here is manually specified the coordinate system with coord_sf. Inside the bracket we've specified **crs**, which stands for **coordinate reference systems**. If you look closely, you can see this map, where CRS = 3857, looks a bit different from the earlier one; it's not night and day, but there's differences. (This is known as pseudo-Mercator.) To get a sense of why they might have more major implicaitons, try the following:

```
h_and_w %>%
  mutate(percent_owned_bin = cut_number(percent_owned, 6)) %>%
  ggplot() +
  geom_sf(aes(fill = percent_owned_bin)) +
  scale_fill_viridis_d(labels =
                         c("less than 55%",
                           "55%-63%",
                           "63-67%",
                           "67-70%",
                           "70-73%",
                           "more than 73%")) +
  labs(fill = "% of households owned
(including with a mortgage)",
       title = "Lots of renters in London") +
  coord_sf(crs = 22275)
```

Why does this look weird?

Task

OK, let's try something different. Please do the following:

- there's another file in this project called **degrees.csv**. This contains data on what fraction of adults in each parliamentary constituency hold degrees. Please load it into R;
- please then merge that file with your existing file, which I'd called h_and_w;
- please draw a map of the fraction of adults in each parliamentary constituency with degrees, treating the DEGREES variable as continuous;
- please draw a map consisting of two colours, where the half the English constituencies (about 286)

where a below-median fraction of adults hold degrees are in one colour, and the other half are in a different colour