Week seven solutions

Just two visualisations in this week's task. But first we have to load our packages, load our data and then merge the **degrees** dataset with the **h_and_w** one.

In the code below, I load the packages and data you used in the lab.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ─
─
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   4.0.0     ✔ tibble    3.3.0
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
## ✔ purrr     1.1.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ─
─
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conf
licts to become errors
```

```
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
westminster <- read_sf(dsn = ".", layer = "westminster_const_region")
housing <- read.csv("housing.csv")
h_and_w <- inner_join(westminster, housing, by = "CODE")
```

Next, I'm going to load the **degrees** dataset.

```
degrees <- read_csv("degrees.csv")
```

```
## Rows: 573 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
─
## Delimiter: ","
## chr (1): CODE
## dbl (1): DEGREES
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messag
e.
```

Finally, I'll use the inner_join function to merge degrees with h_and_w. I'm going to call the new dataset d_and_w. Remember, the inner_join function uses the following format: - Name your new dataset, in this example it is d_and_w - Use the <- operator to tell R we are creating a new object, which will consist of what appears on the right of this arrow - Open the inner_join function - Specify the first dataset to merge, which is h_and_w in this example - Specify the second dataset to merge, which is degrees in this example - Specify the id variable, which is the variable both datasets have in common, so they will merge where the observations in this variable match; in this example the id variable is CODE, which is a unique code for each constituency

```
d_and_w <- inner_join(h_and_w, degrees, by = "CODE")
```

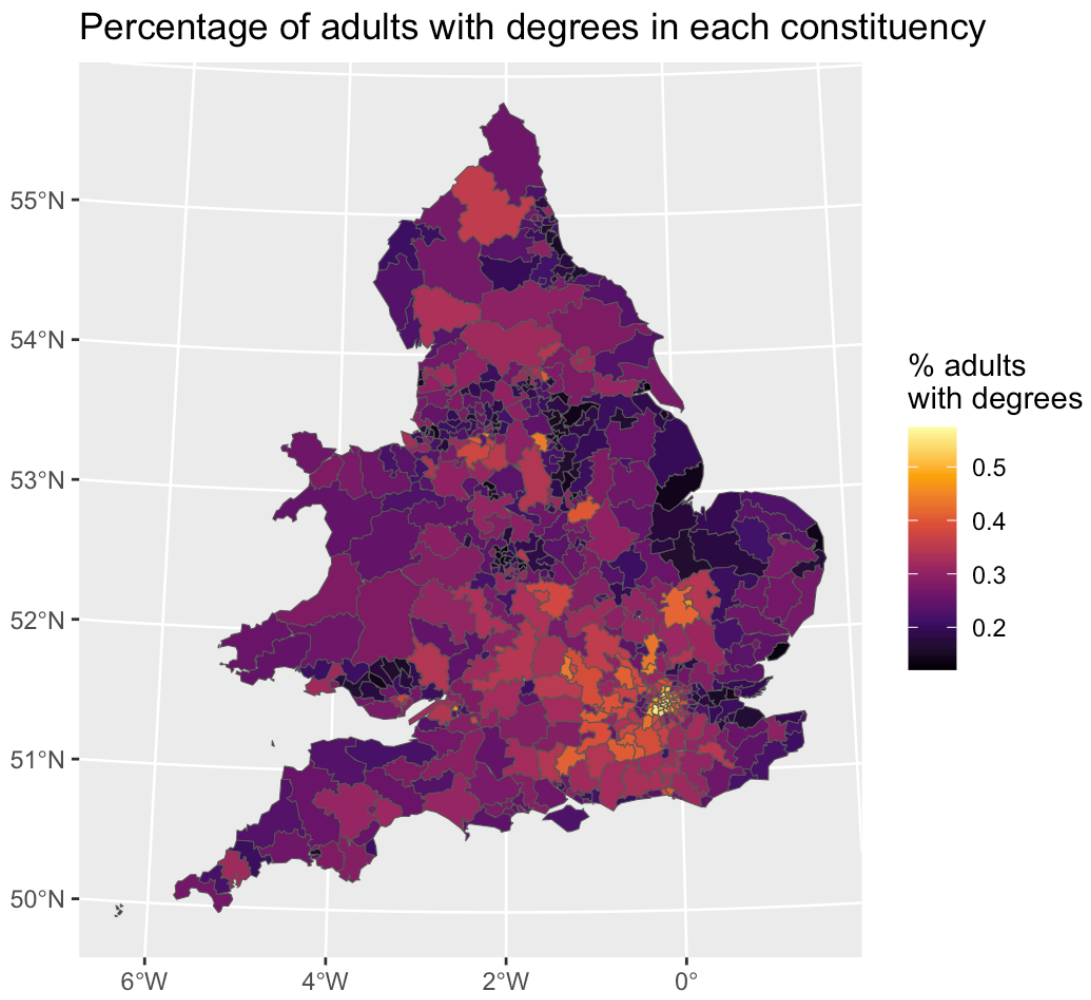
Map 1

Question 1 asks us to produce, "a map of the fraction of adults in each parliamentary constituency with degrees, treating the DEGREES variable as continuous".

Below I've produced the first graph. As we're using a continuous variable for the fill, I've used the function **scale_fill_viridis_c** for the colour scale.

Also, notice that I've used "" within the string that gives the legend a title. This tells R to start a new line at this point in the string, which makes the legend take up less space in the plot.

```
ggplot(data = d_and_w) +
  aes(fill = DEGREES) +
  geom_sf() +
  scale_fill_viridis_c(option = "B") +
  labs(
    title = "Percentage of adults with degrees in each constituency",
    fill = "% adults\nwith degrees"
  )
```



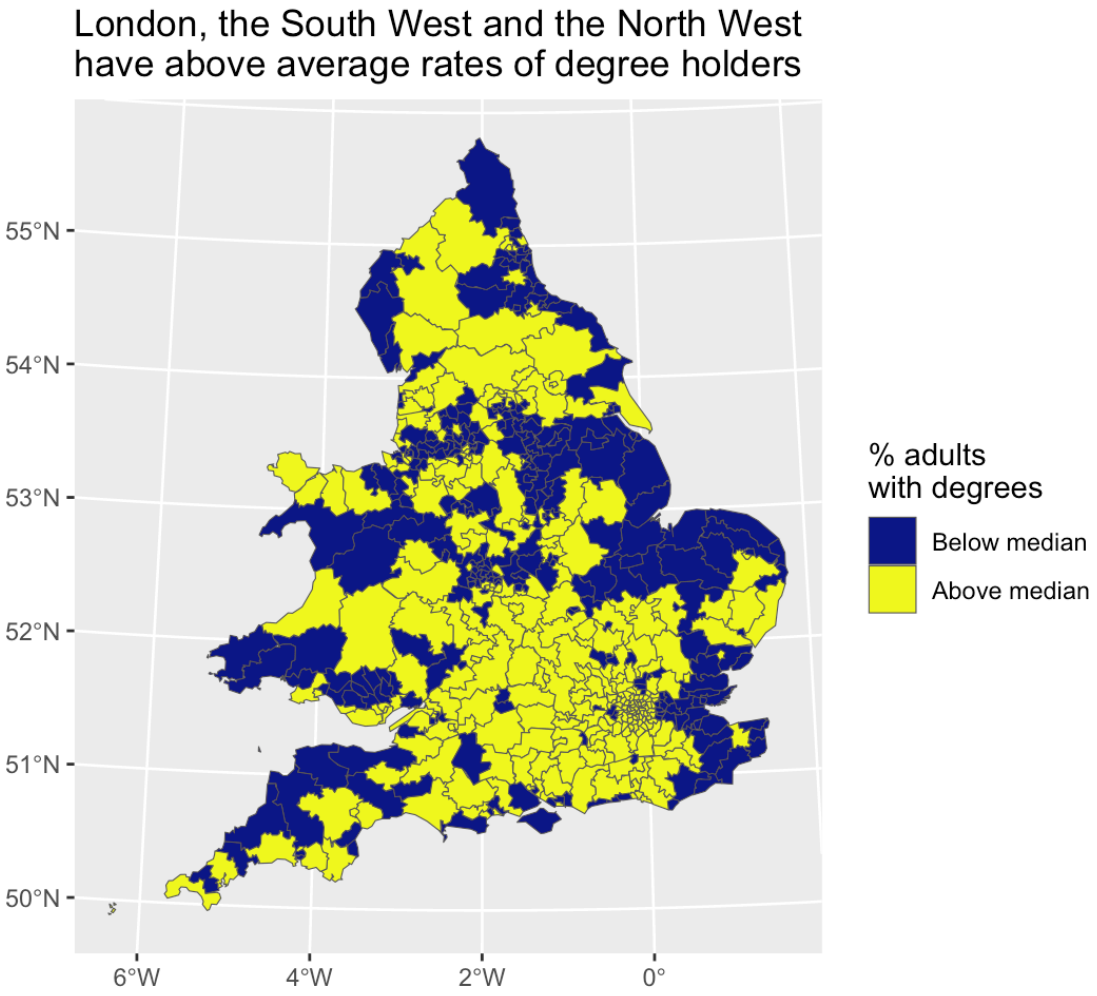
Map 2

Question 2 asks us to produce, “a map consisting of two colours, where 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.”

This map requires you to cut the DEGREES variable into two categories. The categories should have roughly the same amount of observations in each, because we want half in one category and half in the other. The best function to achieve this is the **cut_number** function, because this produces bins that vary in range, but with a roughly equal number of observations in each.

So, let’s use **cut_number** to plot a map with two colours.

```
d_and_w %>%
  mutate(degrees_binned = cut_number(DEGREES, 2)) %>%
  ggplot() +
  aes(fill = degrees_binned) +
  geom_sf() +
  scale_fill_viridis_d(
    option = "C",
    labels = c("Below median", "Above median")
  ) +
  labs(
    title = "London, the South West and the North West\nhave above average rates o
f degree holders",
    fill = "% adults\nwith degrees"
  )
```



In the code above I use the function scale_fill_viridis_d for the colour scale; the d stands for discrete, which is another word for a categorical variable. I've chosen option C in the viridis package for no reason, other than that I like it. You could experiment with any of the options in the function to find one you like; there are four options – A, B, C or D. Finally, I've added labels to the bins in my new variable, and I've added a title to the graph and the legend. Again, I've used "" to tell R to put the titles on multiple lines.

Common mistakes

- The main issue this week was that most people did not submit their tasks to Blackboard; please do

this because it is the main way we give feedback on your graphs, so it is in your interest and will help you learn.

- Using the wrong variable to add colour; some people used the housing variable, not the degrees one. Also make sure you are using scale_**fill** rather than scale_color when in aes() you are using the fill aesthetic.
- Not customising the graph with a title or adding a title to the colour legend — these are things that will be weighted more heavily in assessment 2, so it's good to practice them now!
- Having difficulties installing the sf package; if this happens, please post on Piazza to get a faster reply. Remember that the first time we use a package we need to install it using install.packages()