Week 11 Exercise: Spatial Analysis

Calum Webb

06/12/2021

In this week's practical exercise, you will be re-creating some of the maps shown in the lecture slides and creating your own map for a city of region of interest (other than Sheffield). You will be exploring how house-buying activities in your area of interest have changed over a two-decade period — whether the most popular areas for purchasing homes have changed and whether the most popular home-buying areas have become more concentrated or more dispersed using Moran's I.

Let's start by loading the libraries required.

```
# Remember to install the packages you haven't used before! You should run the install.packages() funct

# install.packages("tidyverse")
# install.packages("sf")
# install.packages("leaflet")
# install.packages("spdep")

library(tidyverse) # For joining and manipulating data
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(sf) # For reading and working with spatial data
```

```
## Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(leaflet) # For creating interactive maps
library(spdep) # For Moran's I statistics
```

```
## Loading required package: sp
## Loading required package: spData
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

Part I: Reading in Spatial Data and joining it to social science data

We are going to start by reading in our spatial data and joining it to some social science data about the small areas (LSOAs). We're then going to filter the data so just one city or region is mapped (as it takes

quite a lot of processing power to create a large map of all 35,000-ish LSOAs!)

We use `st_read` from the `sf` package to read in our shapefile data. This is called `lsoa-england-wales-boundaries` and is saved in the `data` folder. We'll save it to an object called `lsoa_boundaries`.

We'll also read in the LSOA population weighted centroids and social science data, as we'll be using these for calculating Moran's I.

```
# Read in LSOA boundaries
lsoa_boundaries <- st_read("data/lsoa-england-wales-boundaries/")
```

```
## Reading layer `Lower_Layer_Super_Output_Areas_December_2011_Generalised_Clipped__Boundaries_in_Engla
##   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
```

```
# Read in LSOA centroids
lsoa_centroids <- st_read("data/lsoa-england-wales-centroids/")
```

```
## Reading layer `Lower_Layer_Super_Output_Areas_(December_2011)_Population_Weighted_Centroids' from da
##   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
```

```
# Read in additional data (which includes house buying popularity)
housing_data <- read_csv("data/lsoa_data_tidy.csv")
```

```
## Rows: 32844 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): lsoa_code, lsoa_name, utla17nm
## dbl (7): idaopi_2019, percent_eg_rated, housesales_1998, housesales_2018, ho...
##
## 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 message.
```

Next, we'll use the `left_join` function from the `tidyverse` package to join the LSOA housing data onto the LSOA boundaries and LSOA centroids data. We'll call the new merged data `lsoa_boundaries_c` and `lsoa_centroids_c`, where the c stands for "combined"

```
# Join LSOA boundaries and housing data by ID columns lsoa11cd and lsoa_code
lsoa_boundaries_c <- left_join(lsoa_boundaries, housing_data, by = c("lsoa11cd" = "lsoa_code"))

# Join LSOA centroids and housing data by ID columns lsoa11cd and lsoa_code
lsoa_centroids_c <- left_join(lsoa_centroids, housing_data, by = c("lsoa11cd" = "lsoa_code"))
```

Part II: Filtering our data down to a single city/region/county

There are two possible ways we could go about filtering our data down to just one city or region. We could use a simple filter match from the `utla17nm` variable, which contains larger 'upper tier local authority' regions. We can view the names of these in alphabetical order by running the following code:

```r
sort(unique(lsoa_boundaries_c$utla17nm))
```

```
##  [1] "Bath and North East Somerset" "Bedford"
##  [3] "Blackburn with Darwen"        "Blackpool"
##  [5] "Bournemouth"                  "Bracknell Forest"
##  [7] "Brighton and Hove"            "Bristol, City of"
##  [9] "Buckinghamshire"              "Cambridgeshire"
## [11] "Central Bedfordshire"         "Cheshire East"
## [13] "Cheshire West and Chester"    "Cornwall"
## [15] "County Durham"                "Cumbria"
## [17] "Darlington"                   "Derby"
## [19] "Derbyshire"                   "Devon"
## [21] "Dorset"                       "East Riding of Yorkshire"
## [23] "East Sussex"                  "Essex"
## [25] "Gloucestershire"              "Greater Manchester"
## [27] "Halton"                       "Hampshire"
## [29] "Hartlepool"                   "Herefordshire, County of"
## [31] "Hertfordshire"                "Inner London"
## [33] "Isle of Wight"                "Isles of Scilly"
## [35] "Kent"                         "Kingston upon Hull, City of"
## [37] "Lancashire"                   "Leicester"
## [39] "Leicestershire"               "Lincolnshire"
## [41] "Luton"                        "Medway"
## [43] "Merseyside"                   "Middlesbrough"
## [45] "Milton Keynes"                "Norfolk"
## [47] "North East Lincolnshire"      "North Lincolnshire"
## [49] "North Somerset"               "North Yorkshire"
## [51] "Northamptonshire"             "Northumberland"
## [53] "Nottingham"                   "Nottinghamshire"
## [55] "Outer London"                 "Oxfordshire"
## [57] "Peterborough"                 "Plymouth"
## [59] "Poole"                        "Portsmouth"
## [61] "Reading"                      "Redcar and Cleveland"
## [63] "Rutland"                      "Shropshire"
## [65] "Slough"                       "Somerset"
## [67] "South Gloucestershire"        "South Yorkshire"
## [69] "Southampton"                  "Southend-on-Sea"
## [71] "Staffordshire"                "Stockton-on-Tees"
## [73] "Stoke-on-Trent"               "Suffolk"
## [75] "Surrey"                       "Swindon"
## [77] "Telford and Wrekin"           "Thurrock"
## [79] "Torbay"                       "Tyne and Wear"
## [81] "Warrington"                   "Warwickshire"
## [83] "West Berkshire"               "West Midlands"
## [85] "West Sussex"                  "West Yorkshire"
## [87] "Wiltshire"                    "Windsor and Maidenhead"
## [89] "Wokingham"                    "Worcestershire"
## [91] "York"
```

We could then filter the data to include just LSOAs in these upper tier local authority regions by using the dplyr filter function. For example, if we wanted every small area in South Yorkshire, we could use the following code:

```r
syorks_data <- lsoa_boundaries_c %>%
  filter(utla17nm == "South Yorkshire")

syorks_data
```

```
## Simple feature collection with 853 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1.822589 ymin: 53.3016 xmax: -0.8653387 ymax: 53.66119
## Geodetic CRS:  WGS 84
## First 10 features:
##    objectid  lsoa11cd      lsoa11nm     lsoa11nmw st_areasha st_lengths
## 1      7125 E01007317 Barnsley 018A Barnsley 018A   621916.0   4481.639
## 2      7126 E01007318 Barnsley 018B Barnsley 018B   296782.9   4962.064
## 3      7127 E01007319 Barnsley 015A Barnsley 015A  1244328.7   5894.416
## 4      7128 E01007320 Barnsley 018C Barnsley 018C   374213.9   4879.724
## 5      7129 E01007321 Barnsley 015B Barnsley 015B  1753216.4   6923.159
## 6      7130 E01007322 Barnsley 015C Barnsley 015C  1533555.8   9301.341
## 7      7131 E01007323 Barnsley 007A Barnsley 007A   287184.1   2411.426
## 8      7132 E01007324 Barnsley 007B Barnsley 007B   401480.3   4189.841
## 9      7133 E01007325 Barnsley 007C Barnsley 007C   429347.8   3918.106
## 10     7134 E01007326 Barnsley 007D Barnsley 007D   374887.5   4463.110
##        lsoa_name idaopi_2019 percent_eg_rated housesales_1998 housesales_2018
## 1  Barnsley 018A        28.5        11.458333               7              10
## 2  Barnsley 018B        34.0         5.084746               5              25
## 3  Barnsley 015A         6.8        22.058824              19              38
## 4  Barnsley 018C        30.6        18.333333              12              14
## 5  Barnsley 015B         8.1        22.500000              16              36
## 6  Barnsley 015C        22.0        30.864198              25              43
## 7  Barnsley 007A        29.4         6.796117              58              14
## 8  Barnsley 007B        25.8        13.333333              11              29
## 9  Barnsley 007C        30.2        14.545455               8               9
## 10 Barnsley 007D        31.5        12.500000               9              17
##           utla17nm housesales_1998_quantiles housesales_2018_quantiles
## 1  South Yorkshire                         1                         2
## 2  South Yorkshire                         1                        11
## 3  South Yorkshire                         5                        17
## 4  South Yorkshire                         3                         3
## 5  South Yorkshire                         4                        16
## 6  South Yorkshire                         9                        18
## 7  South Yorkshire                        19                         3
## 8  South Yorkshire                         2                        13
## 9  South Yorkshire                         2                         1
## 10 South Yorkshire                         2                         5
##    population                       geometry
## 1        1471 MULTIPOLYGON (((-1.44749 53...
## 2        1644 MULTIPOLYGON (((-1.457235 5...
## 3        1482 MULTIPOLYGON (((-1.431847 5...
## 4        1867 MULTIPOLYGON (((-1.448238 5...
## 5        1457 MULTIPOLYGON (((-1.41932 53...
## 6        2343 MULTIPOLYGON (((-1.452536 5...
## 7        1701 MULTIPOLYGON (((-1.47348 53...
## 8        1790 MULTIPOLYGON (((-1.484543 5...
## 9        1658 MULTIPOLYGON (((-1.47348 53...
```

```
## 10       1661 MULTIPOLYGON (((-1.469826 5...
```

Note that Sheffield is not an option for the `utla17nm` variable — but we can see in the `lsoa11nm` there are clear lower-level place names that we could use. There's no easy way to see all the options, but we could make a sensible guess using the `str_detect` function which is part of tidyverse. Here are a couple of examples:

```
# Boundaries and centroids for Sheffield
sheff_boundaries <- lsoa_boundaries_c %>%
  filter(str_detect(lsoa11nm, "Sheffield"))
sheff_boundaries
```

```
## 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
## First 10 features:
##    objectid  lsoa11cd       lsoa11nm      lsoa11nmw st_areasha st_lengths
## 1      7625 E01007823 Sheffield 069A Sheffield 069A   421798.0   5357.719
## 2      7626 E01007824 Sheffield 066A Sheffield 066A   345345.7   3626.741
## 3      7627 E01007825 Sheffield 066B Sheffield 066B  1264119.4   6425.416
## 4      7628 E01007826 Sheffield 066C Sheffield 066C  1051342.6   6100.298
## 5      7629 E01007827 Sheffield 064A Sheffield 064A  1541136.8   7910.254
## 6      7630 E01007828 Sheffield 059A Sheffield 059A   280466.0   3537.272
## 7      7631 E01007829 Sheffield 059B Sheffield 059B   443268.1   3539.497
## 8      7632 E01007830 Sheffield 064B Sheffield 064B   290064.4   3535.524
## 9      7633 E01007831 Sheffield 059C Sheffield 059C   307751.3   4401.515
## 10     7634 E01007832 Sheffield 059D Sheffield 059D   200169.4   2266.966
##          lsoa_name idaopi_2019 percent_eg_rated housesales_1998 housesales_2018
## 1  Sheffield 069A        16.3         13.84615              28              31
## 2  Sheffield 066A         3.5         32.53012              31              25
## 3  Sheffield 066B         6.1         42.22222              26              33
## 4  Sheffield 066C         3.0         28.91566              31              36
## 5  Sheffield 064A         7.5         33.02752              30              43
## 6  Sheffield 059A        18.0         24.21875              24              38
## 7  Sheffield 059B        10.8         29.05405              38              63
## 8  Sheffield 064B        16.7         12.76596              25              19
## 9  Sheffield 059C        14.0         56.89655              40              37
## 10 Sheffield 059D        11.7         43.39623              48              29
##           utla17nm housesales_1998_quantiles housesales_2018_quantiles
## 1  South Yorkshire                        11                        15
## 2  South Yorkshire                        12                        11
## 3  South Yorkshire                        10                        15
## 4  South Yorkshire                        12                        16
## 5  South Yorkshire                        12                        18
## 6  South Yorkshire                         9                        17
## 7  South Yorkshire                        15                        20
## 8  South Yorkshire                         9                         7
## 9  South Yorkshire                        16                        17
## 10 South Yorkshire                        18                        14
##    population                       geometry
## 1        1678 MULTIPOLYGON (((-1.481237 5...
## 2        1301 MULTIPOLYGON (((-1.48574 53...
## 3        1449 MULTIPOLYGON (((-1.496023 5...
```

```
## 4        1378 MULTIPOLYGON (((-1.506648 5...
## 5        1661 MULTIPOLYGON (((-1.451869 5...
## 6        1724 MULTIPOLYGON (((-1.482062 5...
## 7        1465 MULTIPOLYGON (((-1.483451 5...
## 8        1382 MULTIPOLYGON (((-1.473521 5...
## 9        1539 MULTIPOLYGON (((-1.480063 5...
## 10       1541 MULTIPOLYGON (((-1.472836 5...
```

```
sheff_centroids <- lsoa_centroids_c %>%
  filter(str_detect(lsoa11nm, "Sheffield"))
sheff_centroids
```

```
## 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
## First 10 features:
##    objectid  lsoa11cd       lsoa11nm      lsoa_name idaopi_2019
## 1      6155 E01008066 Sheffield 028B Sheffield 028B        20.4
## 2      6156 E01008067 Sheffield 030D Sheffield 030D        27.7
## 3      6157 E01008064 Sheffield 028A Sheffield 028A        12.0
## 4      6158 E01008062 Sheffield 013E Sheffield 013E        34.0
## 5      6159 E01008063 Sheffield 011B Sheffield 011B        11.2
## 6      6160 E01008060 Sheffield 010D Sheffield 010D        30.5
## 7      6161 E01008061 Sheffield 011A Sheffield 011A        26.5
## 8      6162 E01008068 Sheffield 026B Sheffield 026B        40.8
## 9      6163 E01008069 Sheffield 028C Sheffield 028C        13.7
## 10     6165 E01032585 Sheffield 072E Sheffield 072E         7.2
##    percent_eg_rated housesales_1998 housesales_2018        utla17nm
## 1         53.103448              23              14 South Yorkshire
## 2         62.745098              33              13 South Yorkshire
## 3         55.399061              50              37 South Yorkshire
## 4          7.272727               4               8 South Yorkshire
## 5         10.204082              20              17 South Yorkshire
## 6          5.555556               4              16 South Yorkshire
## 7         10.084034              10              12 South Yorkshire
## 8          0.000000               7              11 South Yorkshire
## 9         54.117647              49              19 South Yorkshire
## 10        15.294118              21              29 South Yorkshire
##    housesales_1998_quantiles housesales_2018_quantiles population
## 1                          8                         4       1513
## 2                         13                         3       1657
## 3                         18                        17       1523
## 4                          1                         1       1714
## 5                          6                         5       1451
## 6                          1                         5       1537
## 7                          2                         3       1696
## 8                          1                         2       4299
## 9                         18                         7       1551
## 10                         7                        14       1359
##                     geometry
## 1  POINT (433535.7 387806.6)
## 2  POINT (433683.6 387664.8)
## 3    POINT (432992.2 387736)
```

```
## 4  POINT (436615.4 391780.3)
## 5    POINT (435927.4 391785)
## 6  POINT (436404.2 392473.2)
## 7  POINT (436145.7 392102.4)
## 8  POINT (434465.2 387552.6)
## 9    POINT (433265.3 387676)
## 10 POINT (443074.4 380751.8)
```

```
# Boundaries and centroids for Torbay
torbay_boundaries <- lsoa_boundaries_c %>%
  filter(str_detect(lsoa11nm, "Torbay"))

torbay_centroids <- lsoa_centroids_c %>%
  filter(str_detect(lsoa11nm, "Torbay"))
```

- Use either of the above options to filter the LSOA boundaries and LSOA centroids data down to a city or region of your choosing. If you can't think of one, try and create a filtered version of the data for all small areas in the 'Inner London' `utla17nm` region.

```
innerldn_boundaries <- lsoa_boundaries_c %>%
  filter(utla17nm == "Inner London")

innerldn_boundaries
```

```
## Simple feature collection with 1895 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -0.25909 ymin: 51.41105 xmax: 0.09674533 ymax: 51.61122
## Geodetic CRS:  WGS 84
## First 10 features:
##    objectid  lsoa11cd            lsoa11nm           lsoa11nmw st_areasha
## 1         1 E01000001 City of London 001A City of London 001A  133320.77
## 2         2 E01000002 City of London 001B City of London 001B  226191.27
## 3         3 E01000003 City of London 001C City of London 001C   57302.97
## 4         4 E01000005 City of London 001E City of London 001E  190738.76
## 5       825 E01000842         Camden 011A         Camden 011A  103698.64
## 6       826 E01000843         Camden 014A         Camden 014A   84528.49
## 7       827 E01000844         Camden 011B         Camden 011B  114746.30
## 8       828 E01000845         Camden 014B         Camden 014B  118424.91
## 9       829 E01000846         Camden 014C         Camden 014C   82062.96
## 10      830 E01000847         Camden 014D         Camden 014D   98828.70
##    st_lengths           lsoa_name idaopi_2019 percent_eg_rated housesales_1998
## 1    2291.846 City of London 001A         1.2        34.210526              82
## 2    2433.960 City of London 001B         3.0        36.538462              47
## 3    1142.360 City of London 001C        12.8        22.543353              65
## 4    2167.868 City of London 001E        32.2         5.479452               8
## 5    2031.620         Camden 011A        11.1        18.032787              43
## 6    1756.106         Camden 014A        13.7        20.202020              43
## 7    2324.477         Camden 011B        13.6        21.875000              52
## 8    2222.829         Camden 014B        12.9        14.000000              51
## 9    1743.392         Camden 014C        24.3        15.873016               6
## 10   1733.056         Camden 014D        19.7         8.602151              31
##    housesales_2018     utla17nm housesales_1998_quantiles
## 1               38 Inner London                        20
## 2               39 Inner London                        18
```

```
## 3               41 Inner London                        20
## 4                5 Inner London                         2
## 5               18 Inner London                        17
## 6               19 Inner London                        17
## 7               22 Inner London                        18
## 8               25 Inner London                        18
## 9                5 Inner London                         1
## 10               6 Inner London                        12
##    housesales_2018_quantiles population                       geometry
## 1                         17       1749 MULTIPOLYGON (((-0.09726264...
## 2                         17       1678 MULTIPOLYGON (((-0.08810299...
## 3                         17       1900 MULTIPOLYGON (((-0.09675966...
## 4                          1       2181 MULTIPOLYGON (((-0.07320468...
## 5                          6       1868 MULTIPOLYGON (((-0.1652361 ...
## 6                          6       1990 MULTIPOLYGON (((-0.1617312 ...
## 7                          9       2169 MULTIPOLYGON (((-0.171989 5...
## 8                         11       1938 MULTIPOLYGON (((-0.1606775 ...
## 9                          1       1729 MULTIPOLYGON (((-0.164945 5...
## 10                         1       2034 MULTIPOLYGON (((-0.1725359 ...
```

```
innerldn_centroids <- lsoa_centroids_c %>%
  filter(utla17nm == "Inner London")

innerldn_centroids
```

```
## Simple feature collection with 1895 features and 12 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 521391.7 ymin: 169899 xmax: 543881.6 ymax: 191625.1
## Projected CRS: OSGB36 / British National Grid
## First 10 features:
##    objectid  lsoa11cd      lsoa11nm     lsoa_name idaopi_2019 percent_eg_rated
## 1     19357 E01002006 Haringey 035B Haringey 035B        11.2         21.49533
## 2     19358 E01002087 Haringey 019D Haringey 019D        36.1         27.39726
## 3     19359 E01002007 Haringey 035C Haringey 035C         9.3         29.78723
## 4     19360 E01002086 Haringey 013B Haringey 013B        31.1         20.93023
## 5     19361 E01002004 Haringey 023D Haringey 023D        30.7         32.00000
## 6     19362 E01002085 Haringey 019C Haringey 019C        39.6         13.29114
## 7     19363 E01002005 Haringey 035A Haringey 035A        23.0         30.12048
## 8     19364 E01002084 Haringey 019B Haringey 019B        28.5         30.00000
## 9     19365 E01002002 Haringey 031C Haringey 031C        36.8         26.22951
## 10    19366 E01002083 Haringey 019A Haringey 019A        31.4         24.00000
##    housesales_1998 housesales_2018     utla17nm housesales_1998_quantiles
## 1               70              21 Inner London                         20
## 2               31              14 Inner London                         12
## 3               38              22 Inner London                         15
## 4               22               9 Inner London                          7
## 5               36              21 Inner London                         15
## 6               36              18 Inner London                         15
## 7               64              23 Inner London                         20
## 8               34              23 Inner London                         14
## 9               30              29 Inner London                         12
## 10              24              11 Inner London                          8
##    housesales_2018_quantiles population                  geometry
## 1                          8       1490   POINT (529203 187601.4)
```

```
## 2                          3       1719 POINT (532032.2 189392.9)
## 3                          9       1510 POINT (528604.3 187937.4)
## 4                          1       1371 POINT (532702.6 189742.2)
## 5                          8       2005   POINT (531501.3 189078)
## 6                          6       2260 POINT (531734.2 189584.7)
## 7                          9       1479 POINT (529047.2 187732.4)
## 8                          9       1767 POINT (532216.8 189675.9)
## 9                         13       1852 POINT (531662.2 188049.8)
## 10                         2       1444 POINT (531940.5 189929.1)
```

Part III: Re-calculating relative popularity of areas for home buying for our filtered data.

At the moment, the `housesales_1998_quantiles` variables are currently reflecting the popularity of house buying in small areas nationally, not locally. We need to re-calcuate them in our filtered data to make them reflect popularity relative to all small areas in the region/city/county. We can simply overwrite them using the `mutate` and `ntile` function. Here is an example using the Sheffield data:

```
sheff_boundaries <- sheff_boundaries %>%
  mutate(
    housesales_1998_quantiles = ntile(x = housesales_1998, n = 20),
    housesales_2018_quantiles = ntile(x = housesales_2018, n = 20)
  )

sheff_centroids <- sheff_centroids %>%
  mutate(
    housesales_1998_quantiles = ntile(x = housesales_1998, n = 20),
    housesales_2018_quantiles = ntile(x = housesales_2018, n = 20)
  )
```

The `ntile` function creates `n` (here, 20) equal sized groups based on the lowest-to-highest values of `x` (here, `housesales_1998`/`housesales_2018`). You can think of these in terms of percentiles. 100 Percentiles divided by 20 equals 5, which means than if a small area's `housesales_1998_quantiles` value is 1 it means it was in the least popular 5% of all of the small areas. Conversely, if a small area's `housesales_1998_quantiles` is 20, it means it was in the most popular 5% of all of the small areas. This can help us make comparisons over time where we might have a general increase or decrease in the range of houses sold.

- Using the above code as a template, re-calculate the `housesales_1998_quantiles` and `housesales_2018_quantiles` for your city or region of interest. (or for Inner London if you can't think of anything)

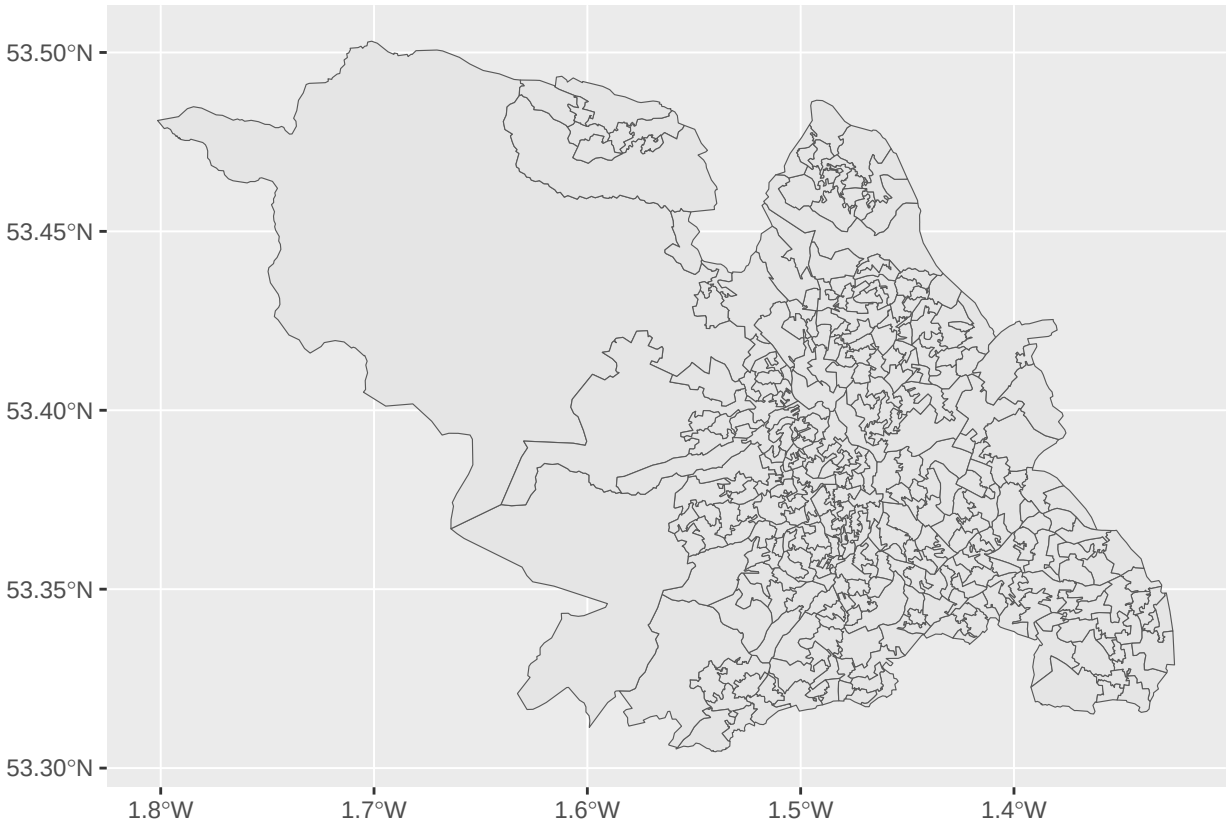
```
innerldn_boundaries <- innerldn_boundaries %>%
  mutate(
    housesales_1998_quantiles = ntile(x = housesales_1998, n = 20),
    housesales_2018_quantiles = ntile(x = housesales_2018, n = 20)
  )

innerldn_centroids <- innerldn_centroids %>%
  mutate(
    housesales_1998_quantiles = ntile(x = housesales_1998, n = 20),
    housesales_2018_quantiles = ntile(x = housesales_2018, n = 20)
  )
```

Part IV: Plotting our data with a (static) choropleth map

Now we can start plotting our spatial data. Let's start by just mapping the 1998 house sales popularity data so that we can adjust our plotting options to make our map readable and visually appealing. We can start with a basic plot of the areas. We use `geom_sf` to plot our spatial data.

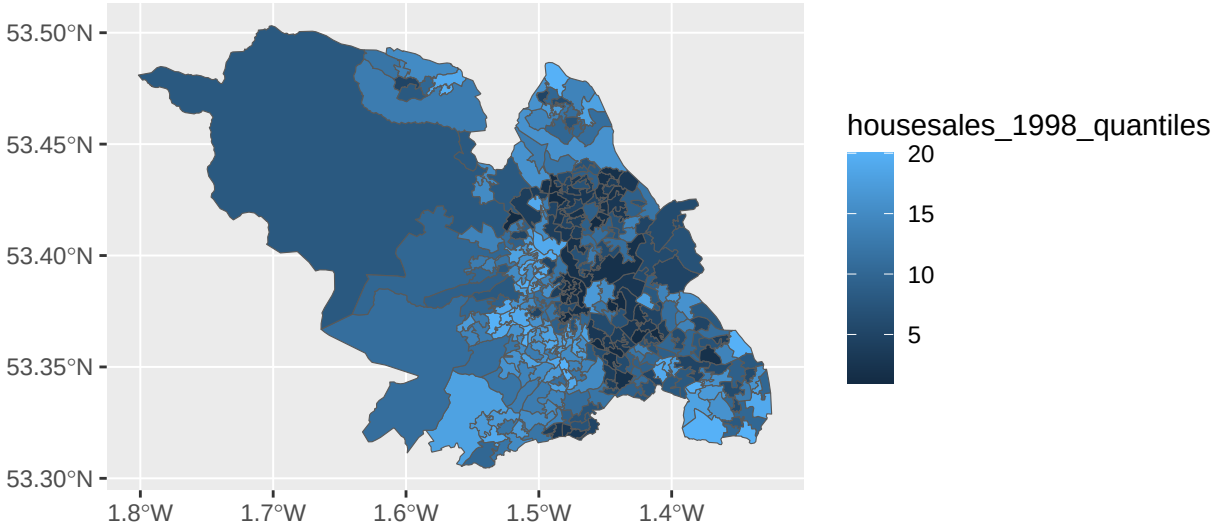
```
sheff_boundaries %>%
  ggplot() +
  geom_sf()
```



The first thing to check is whether the resulting map looks vaguely like what you would expect from the data you've filtered (e.g. it's not the whole map, it doesn't have odd stretches of nowhere and all of the small areas are roughly contiguous (connected)). This looks okay, so we can proceed to fill our spatial areas based on their values of `housesales_1998_quantiles`.

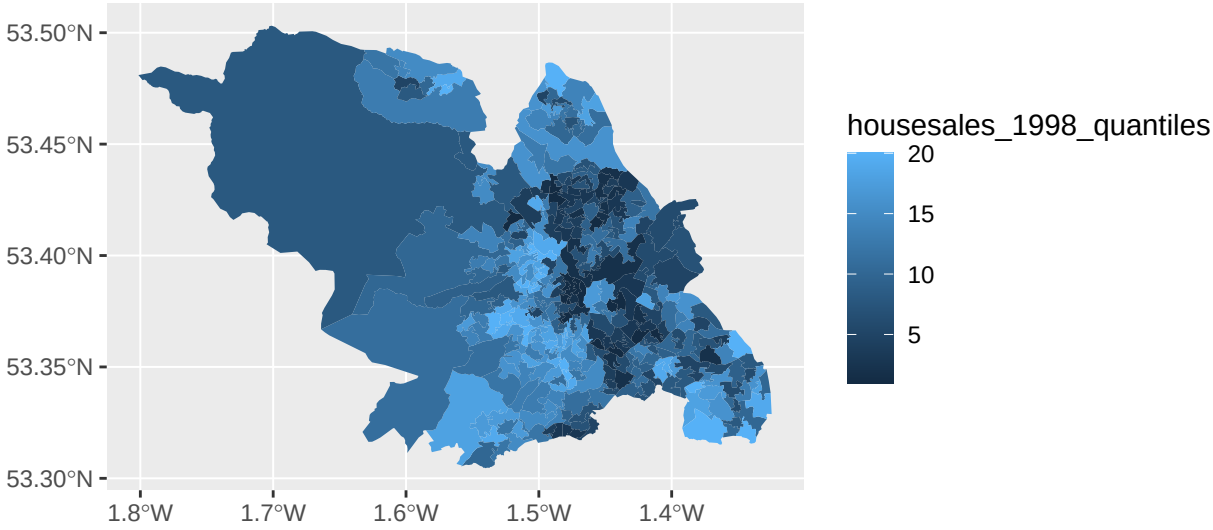
Because we want the values of the fill to be based on a variable, we need to put this within an `aes` function.

```
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles))
```



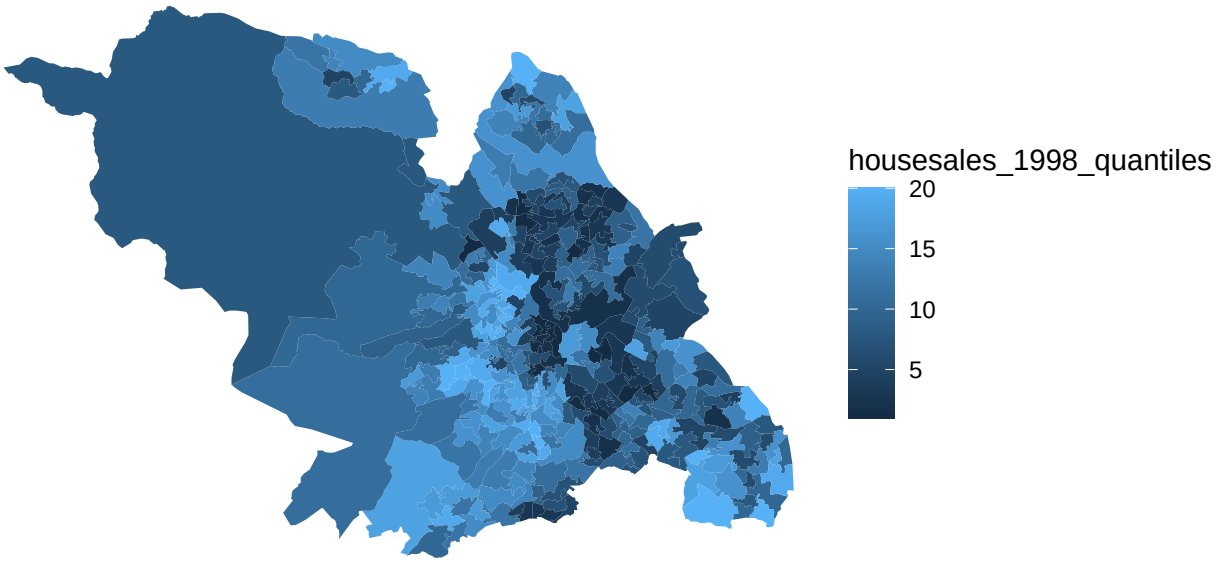
This looks okay, but I personally find the boundary lines to be too intrusive. They obscure a lot of the fill in the more densely populated small areas. We can turn them off by setting the `colour` arhument in `geom_sf` to "transparent" (NB: In `geom_sf` colour/color refers to the colour of the lines, fill refers to the colour inbetween the lines.)

```
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent")
```



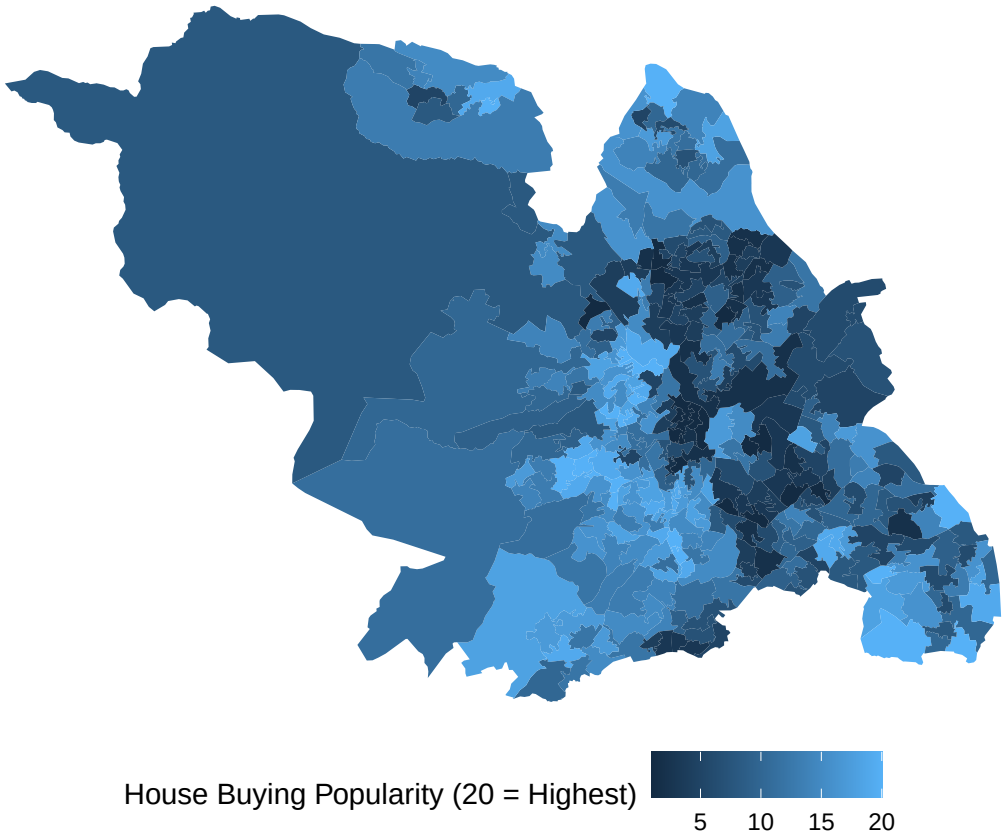
I personally think this looks much better. We could probably improve it further though. We can use the `theme_void()` function to remove all of the grid lines and labels of the plot, since these are not very helpful to us (maybe if we were ship navigators or air traffic controllers!)

```
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent") +
  theme_void()
```



This has the benefit of making our map bigger and more readable! We could do even better if we moved our legend to the top or bottom. We can do this using the `theme()` function. We can also change its title if we wanted.

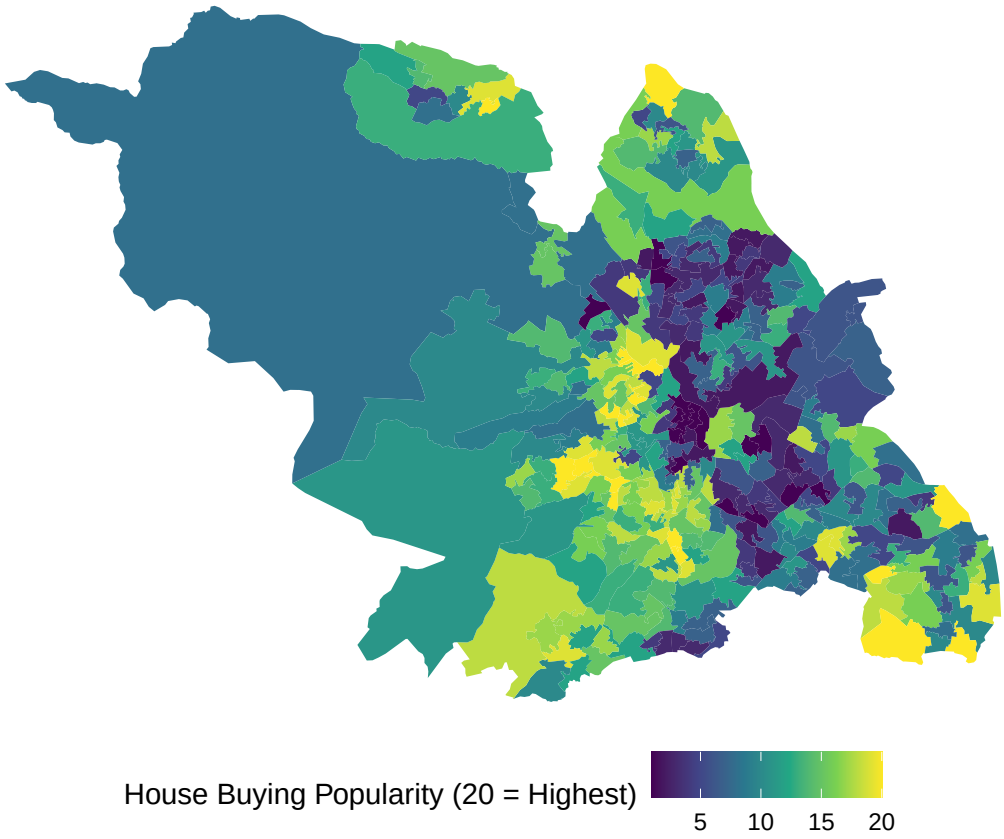
```
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)")
```



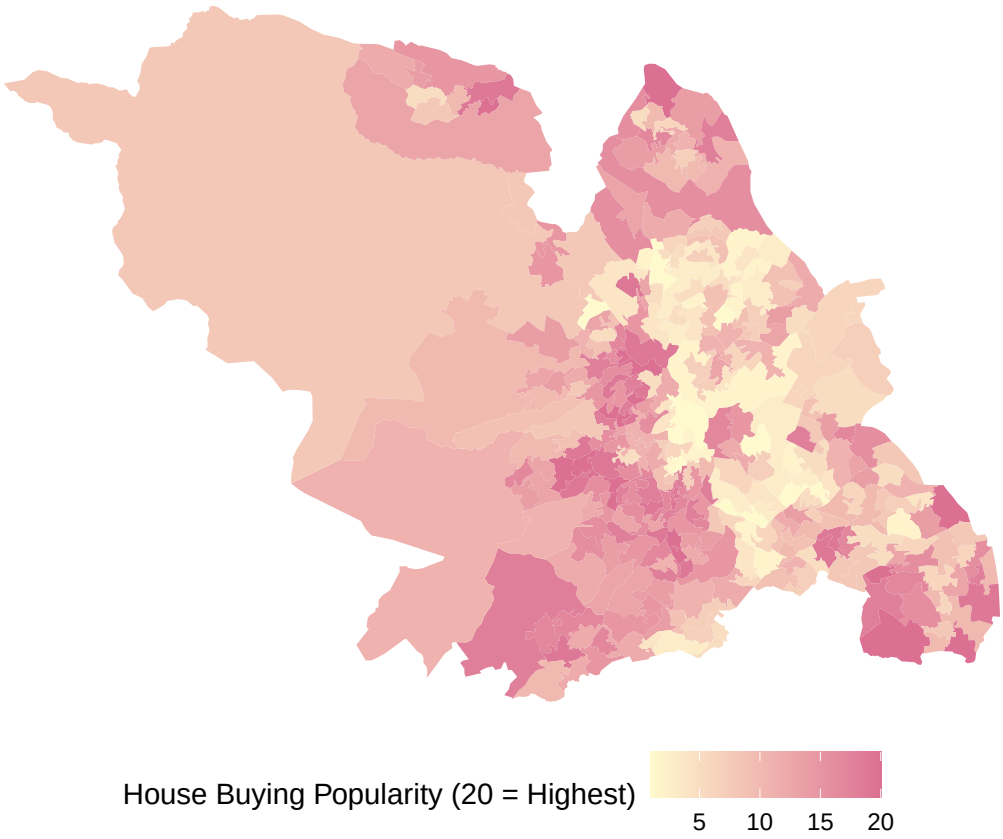
Lastly, we could change the fill colour. One in-built colourblind friendly option is `scale_fill_viridis_c()`, but we could also create our own colour palette using `scale_fill_gradient()` or by using other packages, like the `scico` package which includes scientifically validated packages which avoid distorting variation: https://github.com/thomasp85/scico (https://cran.r-project.org/web/packages/scico/index.html).

Here is a handy chart showing the named colours in `R`: http://sape.inf.usi.ch/sites/default/files/ggplot2-colour-names.png

```
# Colour palette viridis
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)") +
  scale_fill_viridis_c()
```

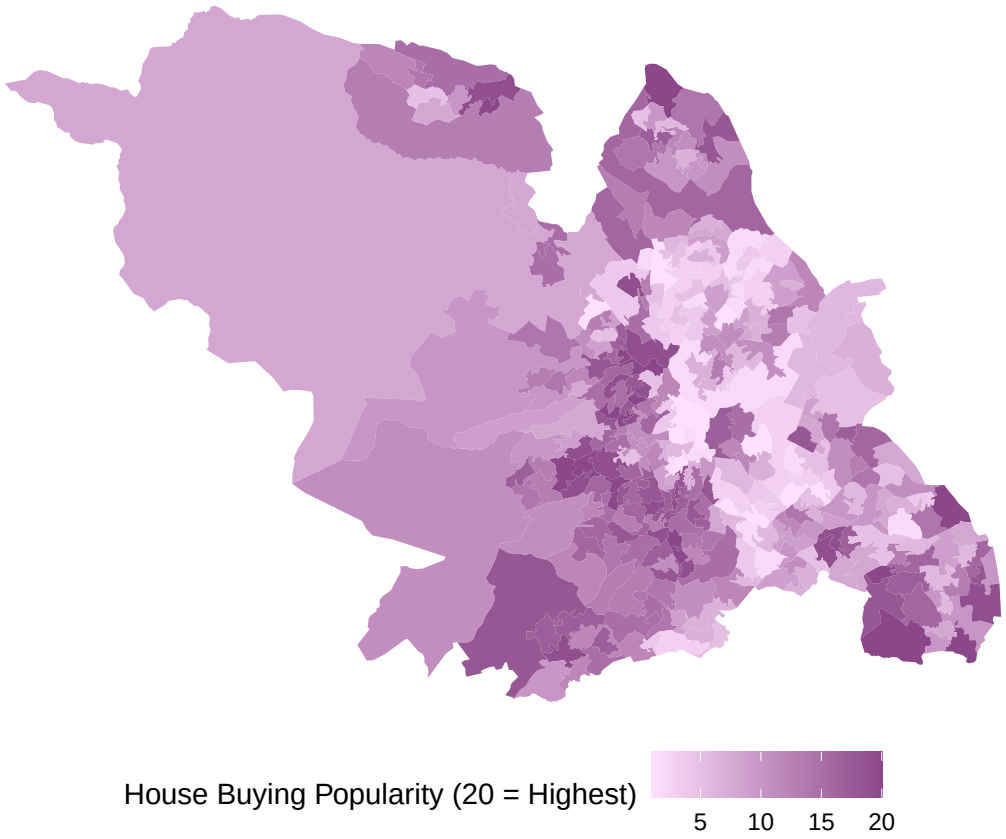



```
# Custom colour palettes using scale_fill_gradient
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)") +
  scale_fill_gradient(low = "lemonchiffon", high = "palevioletred")
```



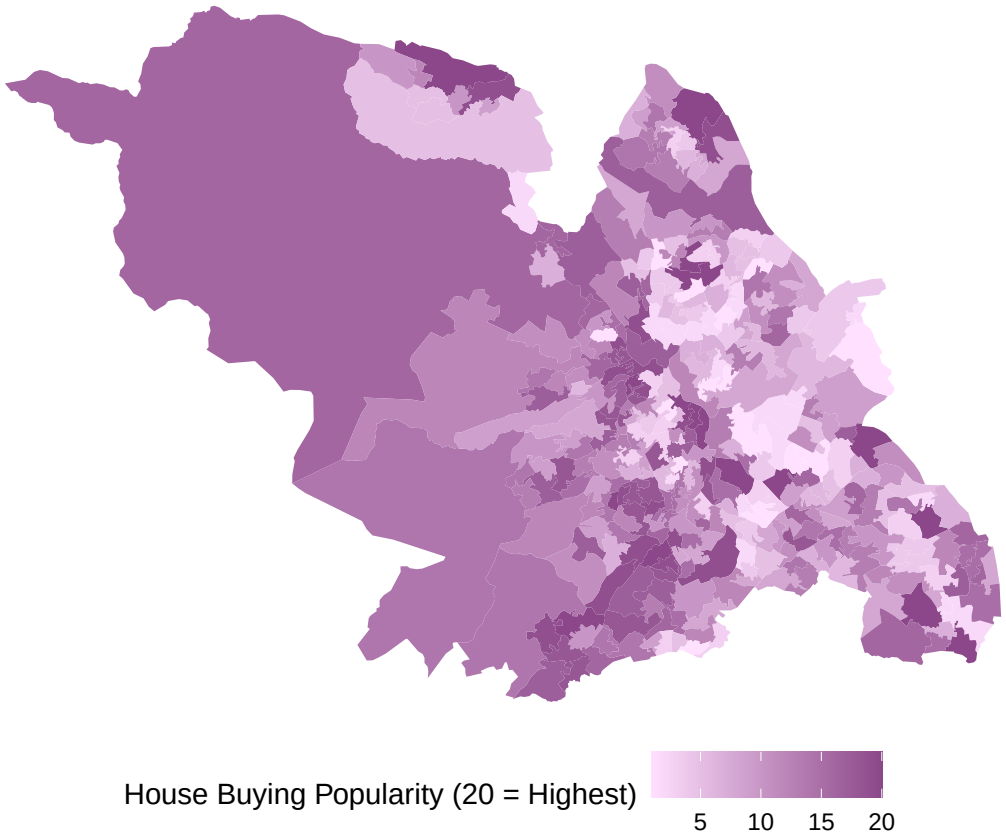
- Try creating your own custom colour palette for the Sheffield Data

```
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)") +
  scale_fill_gradient(low = "thistle1", high = "orchid4")
```



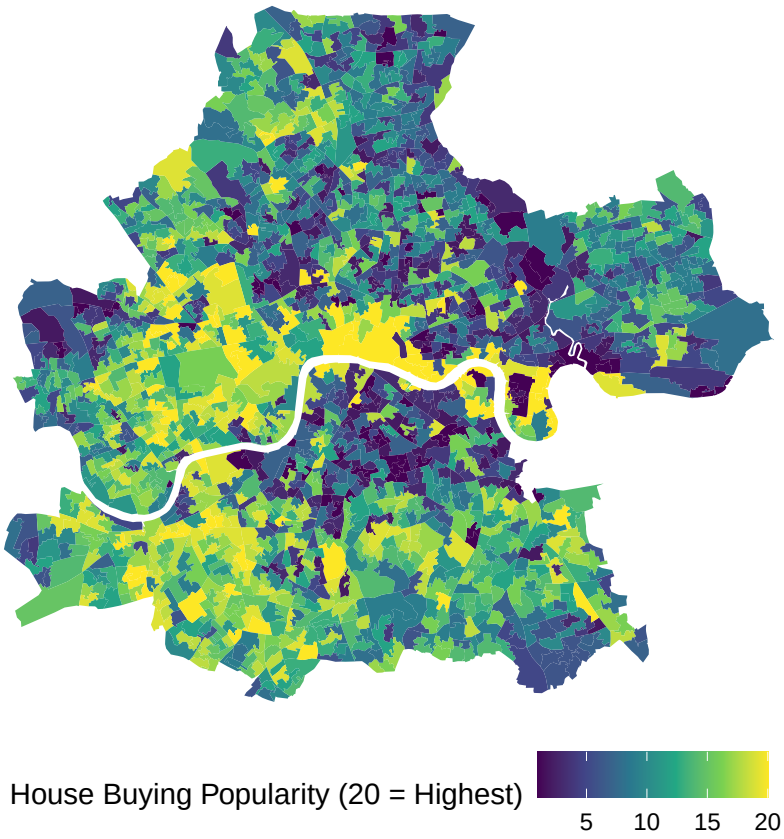
- Now try replicating the plot for Sheffield but mapping the 2018 housebuying quantiles instead

```
sheff_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_2018_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)") +
  scale_fill_gradient(low = "thistle1", high = "orchid4")
```

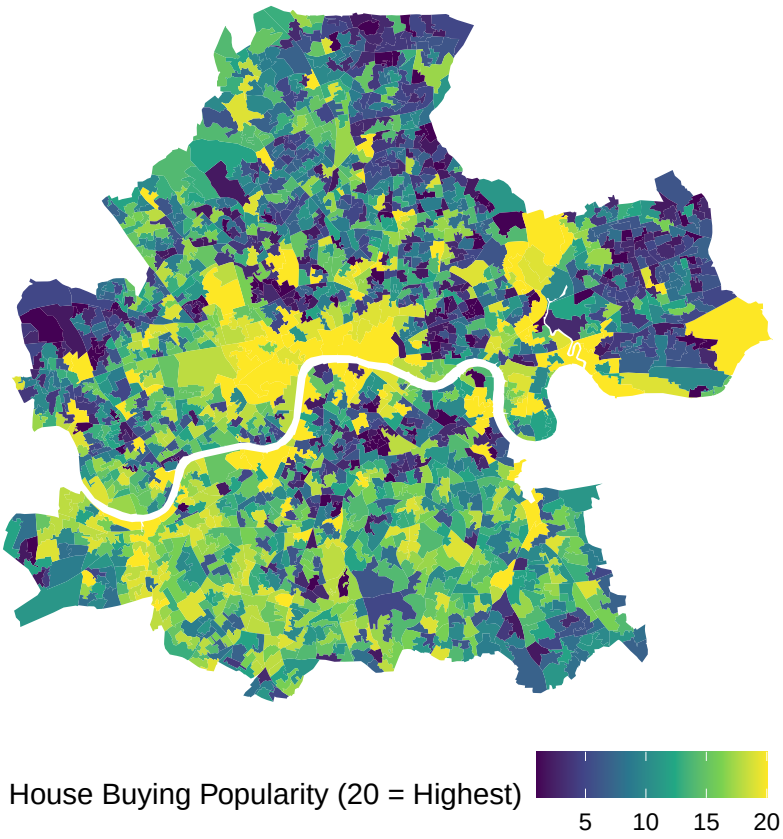


- Finally, try creating a comparable maps for your area of interest — if you couldn't think of one earlier, create a comparable map for Inner London.

```
# 1998
innerldn_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_1998_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)") +
  scale_fill_viridis_c()
```




```
# 2018
innerldn_boundaries %>%
  ggplot() +
  geom_sf(aes(fill = housesales_2018_quantiles), colour = "transparent") +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "House Buying Popularity (20 = Highest)") +
  scale_fill_viridis_c()
```



Part V: Interactive Choropleth Maps

A common complaint of static choropleth maps is that it's hard to distinguish between highly populated areas and, even when it is possible, it can be very difficult for people unfamiliar with maps of the region to identify different areas as there are no markers. Interactive maps that use open source road and area name markers can be very useful here.

One (somewhat) user-friendly option for this in `R` is the `leaflet` package. However, `leaflet` uses very different syntax to what we have seen so far. It also required a longitudinal-latitudinal projection version of the spatial data, which we do not have by default. So, in order to create a `leaflet` interactive map, we need to follow these steps:

- Create a version of our spatial data where it has been transformed to a longlat projection that leaflet can work with
- Create a palette function for our variable (where it's easiest to stick with viridis)
- Create the leaflet plot.

We will work through each step with the Sheffield boundaries data. First, we use the `st_transform` function to convert our data's projection. I'll save the result as `sheff_boundaries_leaflet`. Don't worry too much about what the text in the `crs` argument means.

```
sheff_boundaries_leaflet <- st_transform(x = sheff_boundaries, crs = st_crs("+proj=longlat +datum=WGS84
```

Next, we create a colour palette for our variable/s of interest. Because our variables here are on the same scale (1-20), we only need to create one palette for both and can use either the housesales_1998_quantiles or the housesales_2018_quantiles variable to create it. Here, I've called it `homes_palette`.

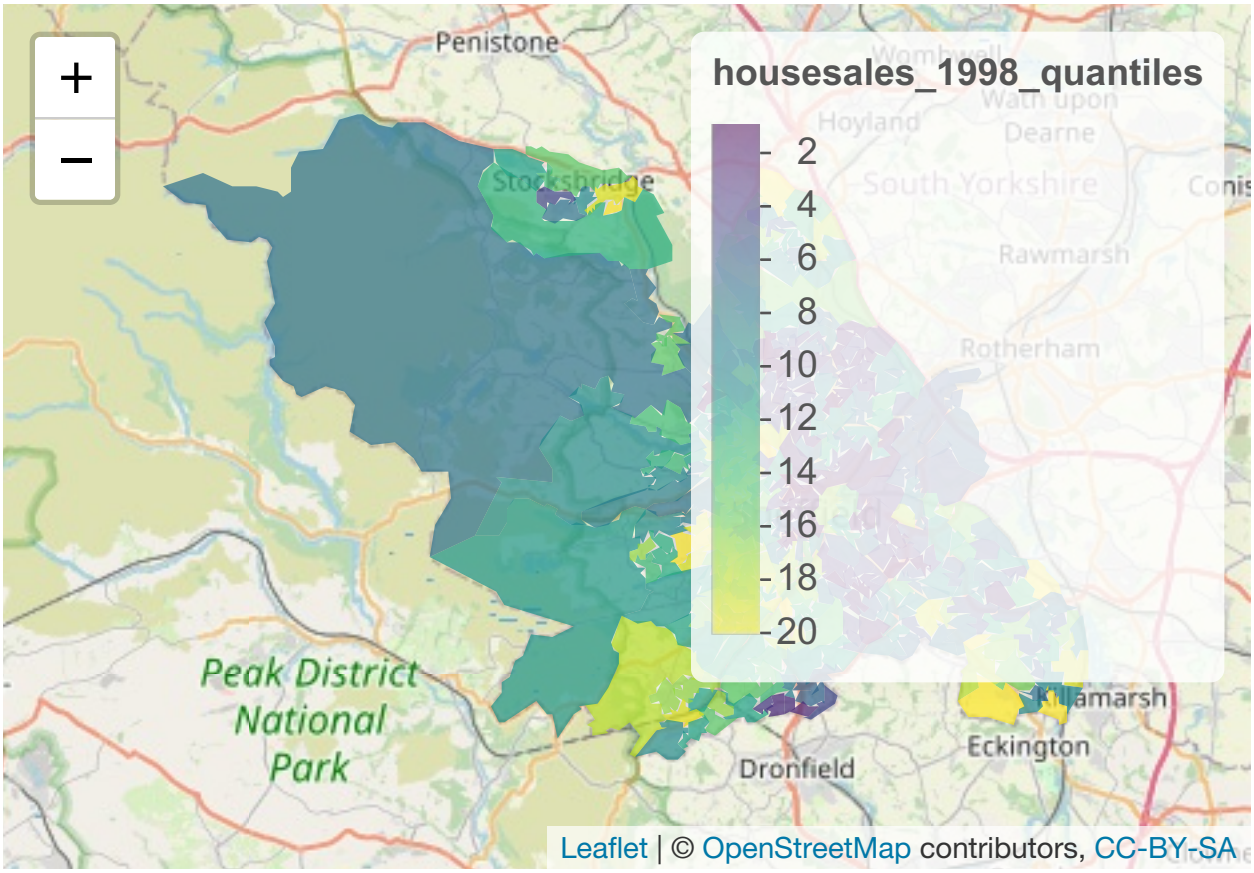
```
homes_palette <- colorNumeric(palette = "viridis",
                              domain = sheff_boundaries_leaflet$housesales_1998_quantiles,
                              na.color = "transparent")
```

Lastly, we can create the `leaflet` plot. When we run this, it will show in our viewer where we could then output it as a html file if we wanted to put it online.

To explain what each part of the code means:

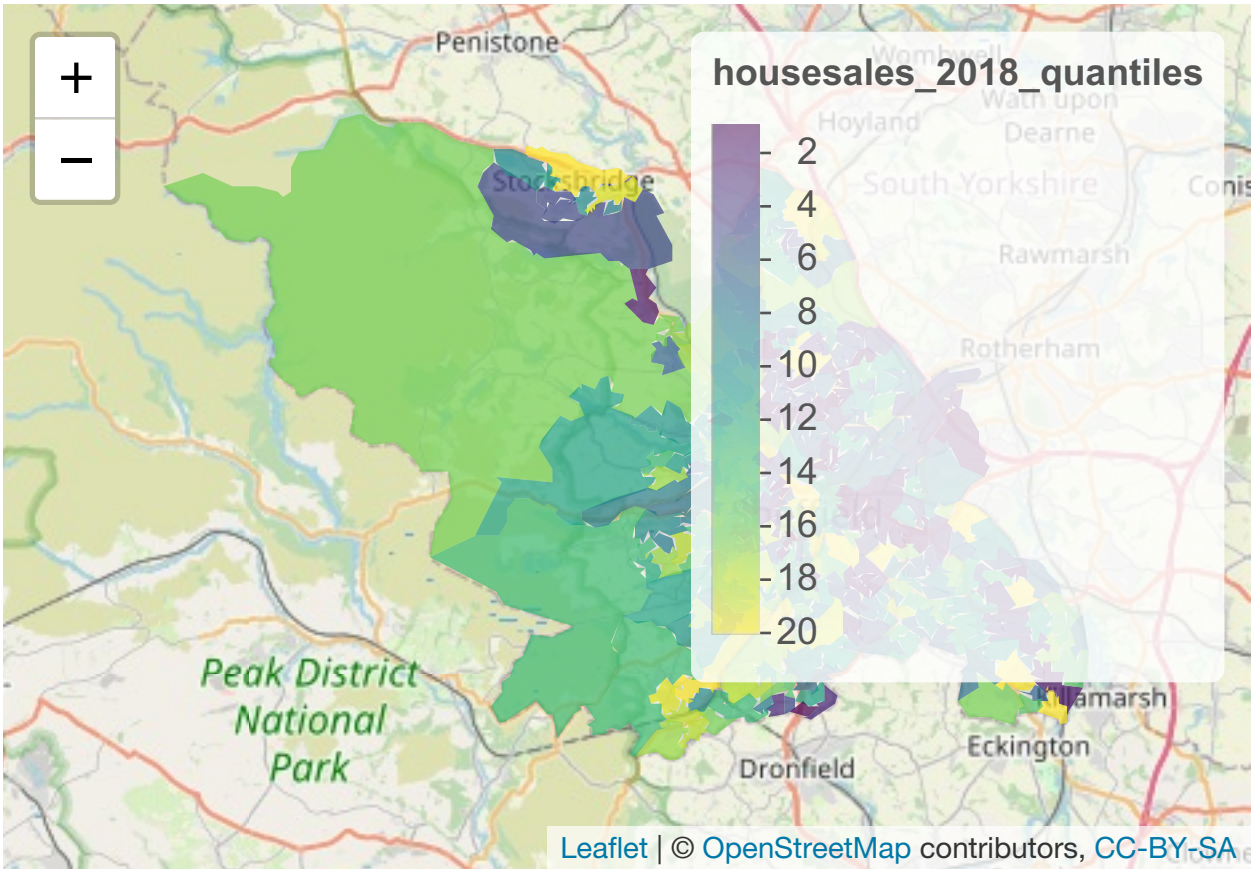
- `leaflet(data = sheff_boundaries_leaflet) %>%` First, we call 'leaflet' to tell R we want to create an interactive leaflet map. We then use the pipe (%>%) for the next argument.
- `addPolygons(stroke = FALSE, fillColor = ~homes_palette(housesales_1998_quantiles), fillOpacity = 0.7) %>%` This part of the code is used for drawing the boundaries (polygons). The option `stroke = FALSE` refers to us telling leaflet to not draw lines around our polygons; `fillColor = ~homes_palette(housesales_1998_quantiles)` tells R we want to use the homes_palette function we created earlier to fill our polygons according to their value in the `housesales_1998_quantiles` variable; `fillOpacity = 0.7` tells leaflet how much transparency we want our polygon fills to have — we want them to be filled solid enough for us to clearly be able to tell the difference between popular and unpopular areas, but not so solid that we cannot read the place names underneath when we zoom in. You may want to adjust this to your liking!
- `addTiles() %>%` adds the underlying map information from OpenStreetMap — this is where the place names, roads, parks, etc. come from.
- `addLegend(pal = homes_palette, values = ~housesales_1998_quantiles)` finally, `addLegend` adds the legend to explain the colour coding. `pal = homes_palette` tells leaflet we want to use the palette we created earlier, `values = ~housesales_1998_quantiles` reiterates to leaflet that we want to base the range on the `housesales_1998_quantiles` variable — this might seem redundant but it's just because we are used to the convenience of ggplot!

```
leaflet(data = sheff_boundaries_leaflet) %>%
  addPolygons(stroke = FALSE,
              fillColor = ~homes_palette(housesales_1998_quantiles),
              fillOpacity = 0.7) %>%
  addTiles() %>%
  addLegend(pal = homes_palette,
            values = ~housesales_1998_quantiles)
```



- Okay, before you try creating your own interactive map for your area of choice, try changing the above code to create a leaflet map for the `housesales_2018_quantiles` variable instead of the 1998 version.

```
leaflet(data = sheff_boundaries_leaflet) %>%
  addPolygons(stroke = FALSE,
              fillColor = ~homes_palette(housesales_2018_quantiles),
              fillOpacity = 0.7) %>%
  addTiles() %>%
  addLegend(pal = homes_palette,
            values = ~housesales_2018_quantiles)
```

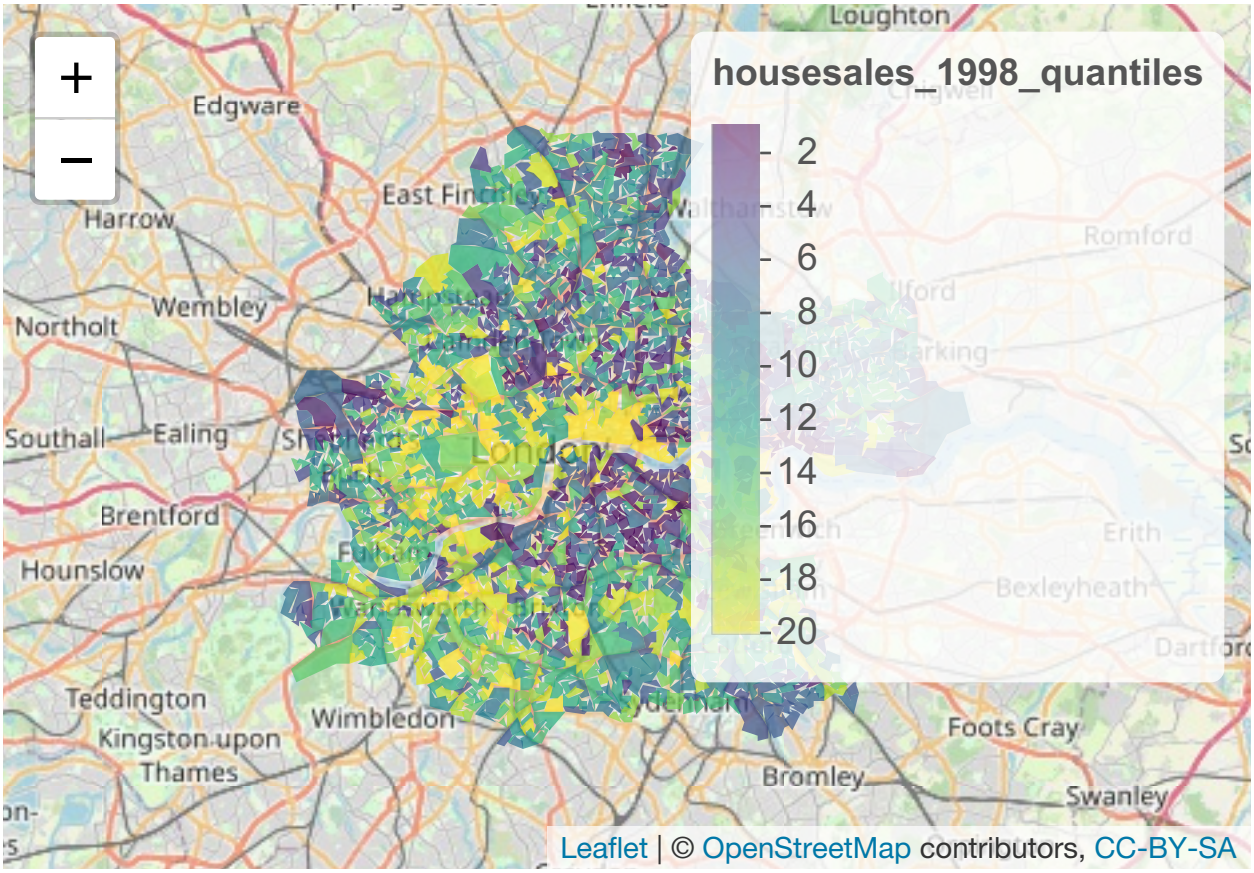


- Now, try following the three steps above to create the leaflet maps for your areas of interest. Again, if you couldn't decide on one, do this for the Inner London region.

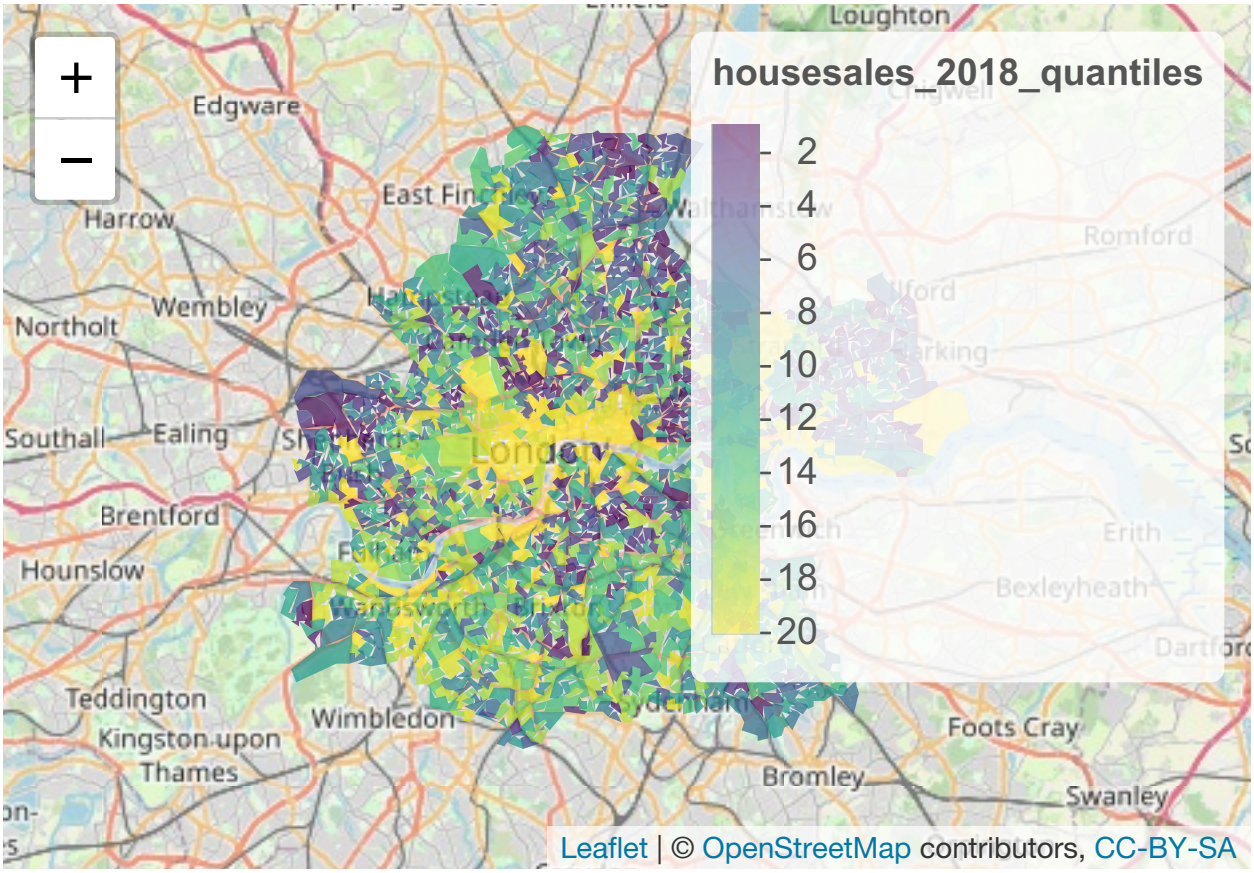
```
london_boundaries_leaflet <- st_transform(x = innerldn_boundaries, crs = CRS("+proj=longlat +datum=WGS84

homes_palette <- colorNumeric(palette = "viridis",
                              domain = london_boundaries_leaflet$housesales_1998_quantiles,
                              na.color = "transparent")

# 1998
leaflet(data = london_boundaries_leaflet) %>%
  addPolygons(stroke = FALSE,
              fillColor = ~homes_palette(housesales_1998_quantiles),
              fillOpacity = 0.7) %>%
  addTiles() %>%
  addLegend(pal = homes_palette,
            values = ~housesales_1998_quantiles)
```




```
# 2018
leaflet(data = london_boundaries_leaflet) %>%
  addPolygons(stroke = FALSE,
              fillColor = ~homes_palette(housesales_2018_quantiles),
              fillOpacity = 0.7) %>%
  addTiles() %>%
  addLegend(pal = homes_palette,
            values = ~housesales_2018_quantiles)
```



Part VI: Moran's I

Lastly, we can calculate our Moran's I using our centroid data. Just like using `leaflet`, it can be a little tricky to calculate Moran's I in `R`, but we can break it down into a few steps:

- Get our centroid coordinates using `st_coordinates` from the `sf` package.
- Use these to calculate a Euclidean distance matrix, using the built in `dist` function. We also need to change it to formally be a `matrix` object using `as.matrix()`
- Invert this `matrix` to create a closeness matrix scaled between 0 and 1 by dividing 1 by every value in the Euclidean distance matrix. I then also need to replace all of the diagonals in the matrix with 0 because they will be 0 by default and result in 1/0 which = Infinity in R.
- Then I can run the Moran's I test using `moran.test` from the `spdep` package.

First, I'll get my coordinates for Sheffield's centroids:

```
sheff_coords <- st_coordinates(sheff_centroids)
```

Then, I'll calculate a distance matrix using `dist()` and convert it into a true matrix object using `as.matrix`

```
sheff_dist <- dist(sheff_coords)
sheff_dist <- as.matrix(sheff_dist)
```

Now I can scale and invert the matrix using the following code:

```
# scale/invert matrix
sheff_close <- 1/sheff_dist
# set diagonals to 0
```

```
diag(sheff_close) <- 0
```

Now I can run the Moran's I test using the `moran.test` function. It can also be helpful to change the scientific notation rules when using this function. For example, if I run `options(scipen = 10)` first, it will mean that scientific notation will only be used when the number printed by `R` is greater than 10 digits.

- `x` = our variable of interest, which we hypothesise may be spatially clustered or dispersed
- `listw` = our scaled closeness matrix.
- `mat2listw` = A convenience function that turns our matrix object into a `listw` object specific to use by `spdep`

Be aware that the Moran's I test can take a little while to run!

```
options(scipen = 10)

moran.test(x = sheff_centroids$housesales_1998_quantiles,
           listw = mat2listw(sheff_close))
```

```
##
##  Moran I test under randomisation
##
## data:  sheff_centroids$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
```

If I wanted to calculate the Moran's I for any other variable in the data, all I would need to do is change the `x` argument in the `moran.test` function. I do not need to recreate the spatial closeness/distance matrix.

For example, to get the Moran's I for the 2018 housing sales quantiles I would only need to change the code to this:

```
moran.test(x = sheff_centroids$housesales_2018_quantiles,
           listw = mat2listw(sheff_close))
```

```
##
##  Moran I test under randomisation
##
## data:  sheff_centroids$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
```

- Follow the above steps, using the template code, to calculate the Moran's I for the house buying popularity variables in the area you chose. If you couldn't think of an area, you could do this for Inner London.

```
london_coords <- st_coordinates(innerldn_centroids)

london_dist <- dist(london_coords)
```

```r
london_dist <- as.matrix(london_dist)

# scale/invert matrix
london_close <- 1/london_dist
# set diagonals to 0
diag(london_close) <- 0

# 1998 - warning: takes a long time to run
moran.test(x = innerldn_centroids$housesales_1998_quantiles,
           listw = mat2listw(london_close))
```

```
##
##  Moran I test under randomisation
##
## data:  innerldn_centroids$housesales_1998_quantiles
## weights: mat2listw(london_close)
##
## Moran I statistic standard deviate = 74.849, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##   0.0709241322939  -0.0005279831045   0.0000009112961
```

```r
# Moran's I = 0.071

# 2018 - warning: takes a long time to run
moran.test(x = innerldn_centroids$housesales_2018_quantiles,
           listw = mat2listw(london_close))
```

```
##
##  Moran I test under randomisation
##
## data:  innerldn_centroids$housesales_2018_quantiles
## weights: mat2listw(london_close)
##
## Moran I statistic standard deviate = 47.821, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##   0.0451228275973  -0.0005279831045   0.0000009112961
```

```r
# Moran's I = 0.045
```

- Did the popular areas for home buying become more or less clustered after 20 years in your example? Or were they approximately the same degree of clustered/dispersed (though the location of the clusters may have changed).

In Inner London, the spatial concentration of popular areas for buying property decreased slightly between the two years. In 1998, the Moran's I statistic for Inner London was approximately 0.071 ($p<0.05$), indicating some significant clustering of popular small areas for property, but this was quite weak and close to random distribution in space. In 2018, the Moran's I statistic for Inner London was approximately 0.045, indicating further weakening of any spatial concentration.

Week 11 Challenge

- Map, and calculate a Moran's I statistic for the proportion of older people living in income deprivation (`idaopi_2019`) in the area you chose.
- Try and read in, join, and plot the proportion of people voting to leave the EU in each Westminster Constituency on the constituency cartogram developed by the House of Commons Library (`constituencies-cartogram.gpkg`). You may need to check slides 21 to 24 for how to work with geopackage files.
- Try estimating a spatial autocorrelation model using the code on slide 64 as a template. Look at the linear association between house buying popularity quantile in 1998 (independent variable) and house buying popularity quantile in 2018 (dependent variable), before and after adjusting for spatial autocorrelation using a spatial lag regression model with the `spaMM` package. Warning: This is a difficult challenge and spatial lag regression models can take a while to estimate!