Maps in R

A very simple introduction

Ben Anderson & Tom Rushby (ECCD)

Last run at: 2021-06-21 15:14:32

Contents

1 R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunks

We like R Markdown. In this file we step through some very simple mapping functions in R/Rmd.

2 Geo-comp

Doing maps and stuff in R.

We use https://geocompr.robinlovelace.net/ a lot...

3 Explore 'World' maps

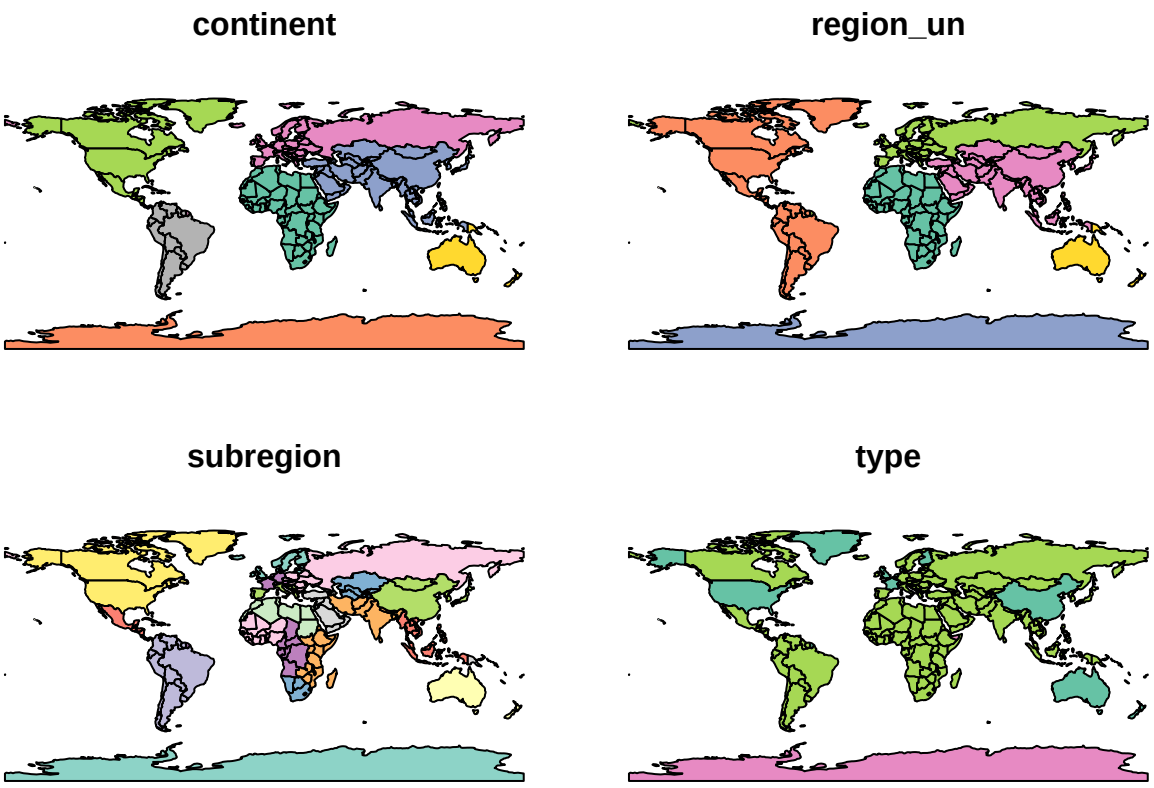
First try using the `sf` package's built-in `world` data to make a simple map plot.

It's worth remembering that a map is just a plot with spatial arrangements...

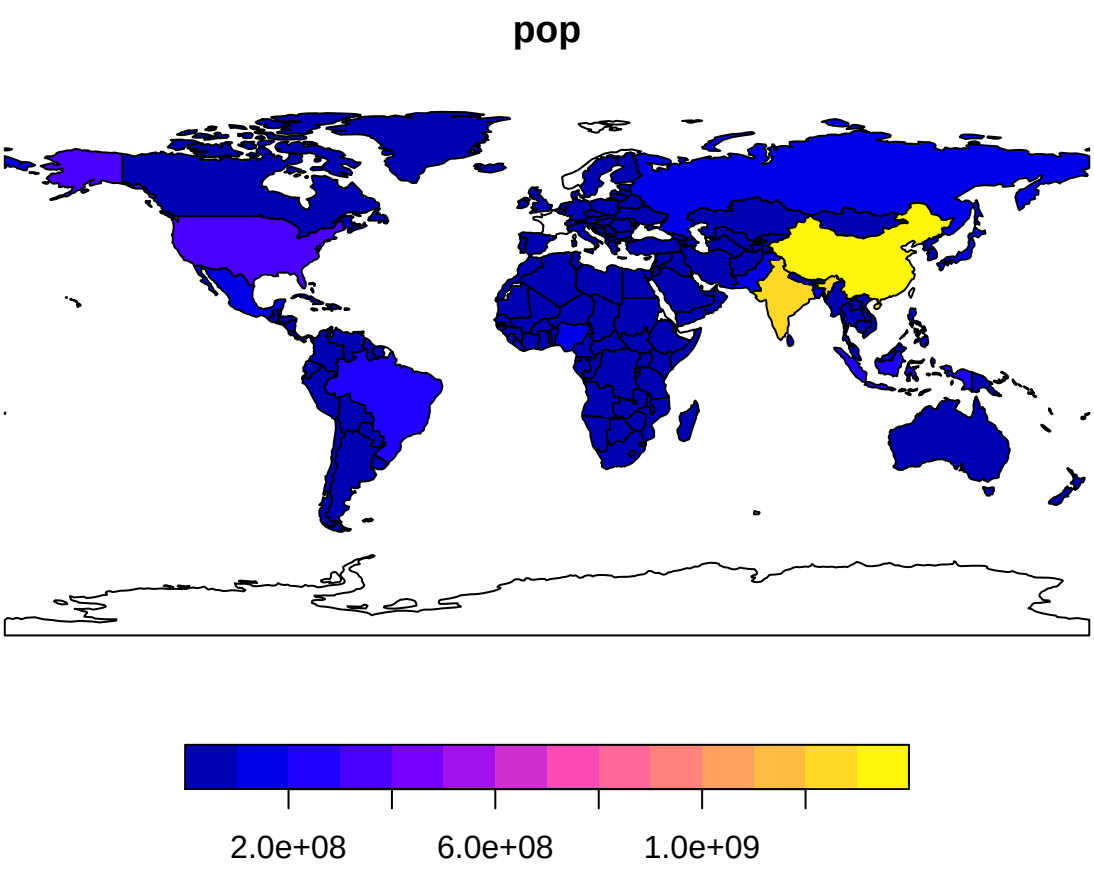
```
head(world)
```

```
## Simple feature collection with 6 features and 10 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -18.28799 xmax: 180 ymax: 83.23324
## Geodetic CRS:  WGS 84
## # A tibble: 6 x 11
##   iso_a2 name_long  continent region_un subregion type  area_km2     pop lifeExp
##   <chr>  <chr>      <chr>     <chr>     <chr>     <chr>    <dbl>   <dbl>   <dbl>
## 1 FJ     Fiji       Oceania   Oceania   Melanesia Sove~   1.93e4  8.86e5    70.0
## 2 TZ     Tanzania   Africa    Africa    Eastern ~ Sove~   9.33e5  5.22e7    64.2
## 3 EH     Western S~ Africa    Africa    Northern~ Inde~   9.63e4 NA         NA
## 4 CA     Canada     North Am~ Americas  Northern~ Sove~   1.00e7  3.55e7    82.0
## 5 US     United St~ North Am~ Americas  Northern~ Coun~   9.51e6  3.19e8    78.8
## 6 KZ     Kazakhstan Asia      Asia      Central ~ Sove~   2.73e6  1.73e7    71.6
## # ... with 2 more variables: gdpPercap <dbl>, geom <MULTIPOLYGON [°]>
```

```
# plot variables 3 to 6
plot(world[3:6])
```

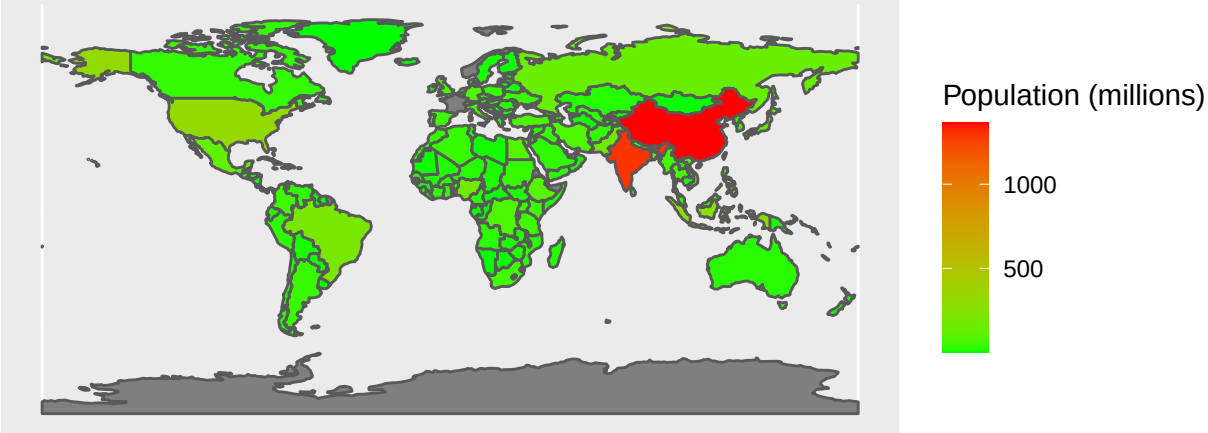


```
# plot world population
plot(world["pop"])
```



Now re-draw the population map using `ggplot2` and the `geom_sf` geometry... This will make a much prettier map because `ggplot2` is so cool.

```
# using ggplot
library(ggplot2)
ggplot2::ggplot(world) +
  geom_sf(aes(fill = pop/1000000)) +
  scale_fill_continuous(name="Population (millions)",
                        low = "green",
                        high = "red")
```



That looks better... well, if we sorted out the colours and the visual dominance of 2 countries...

4 Local Authorities

4.1 Loading data

Now we're going to load some polygon boundary data (you need internet access for this) and some energy demand data so we can link them and map them.

```
library(readr)

# electricity consumption data at LA level
la_elecData <- readr::read_csv("https://assets.publishing.service.gov.uk/government/uploads/system/uploa
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   Region = col_character(),
##   `Local Authority` = col_character(),
##   `LA Code` = col_character(),
##   LAU1 = col_character()
## )
## i Use `spec()` for the full column specifications.
```

This is electricity consumption data for 2019 for English Local Authorities. What variables have we got?

```
head(la_elecData)
```

```
## # A tibble: 6 x 25
##   Region     `Local Authority`    `LA Code` LAU1    E7_meters Standard_meters
##   <chr>      <chr>                <chr>     <chr>       <dbl>           <dbl>
## 1 North East Hartlepool           E06000001 UKC1101      1.82            42.1
## 2 North East Middlesbrough        E06000002 UKC1201      3.13            60.1
## 3 North East Redcar and Cleveland E06000003 UKC1202      3.07            62.0
## 4 North East Stockton-on-Tees     E06000004 UKC1102      3.51            84.4
```

```
## 5 North East Darlington           E06000005 UKC1300      3.12            48.3
## 6 North East County Durham        E06000047 UKC1400      7.71           238.
## # ... with 19 more variables: All_domestic_meters <dbl>,
## #   All_Non-domestic_meters <dbl>, Total_meters <dbl>, E7_GWh <dbl>,
## #   Standard_GWh <dbl>, All_domestic_GWh <dbl>, All_non-domestic_GWh <dbl>,
## #   Total_GWh <dbl>, E7_Mean_kWh <dbl>, E7_Mediann_kWh <dbl>,
## #   Standard_Mean_kWh <dbl>, Standard_Median_kWh <dbl>,
## #   Domestic_Mean_kWh <dbl>, Domestic_Median_kWh <dbl>,
## #   Non-domestic_Mean_kWh <dbl>, Non-domestic_Median_kWh <dbl>,
## #   Total_Mean_kWh <dbl>, Total_Median_kWh <dbl>,
## #   Average_household_mean_kWh <dbl>
```

Now load the boundary data - we will use Local Authority boundaries for the Solent region only to keep things small & easy to play with.

```
# las_to_load <- c("Southampton","Portsmouth","Winchester",
#                  "Eastleigh","Isle of Wight","Fareham",
#                  "Gosport","Test Valley","East Hampshire",
#                  "Havant","New Forest","Hart","Basingstoke and Deane")

# we pre-saved this data in the data folder of the repo for speed
# see getBoundaryData.R for how it works

inf <- here::here("data", "boundaries", "la_solent.shp") # use here to specify the data location
message("Loading LA geometry from ONS Open Geography API")
```

```
## Loading LA geometry from ONS Open Geography API
```

```
la_sf_data <- sf::read_sf(inf)


head(la_sf_data)
```

```
## Simple feature collection with 6 features and 4 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1.586543 ymin: 50.57491 xmax: -0.9747962 ymax: 51.38392
## Geodetic CRS:  WGS 84
## # A tibble: 6 x 5
##   lad18cd  lad18nm       long   lat                                   geometry
##   <chr>    <chr>        <dbl> <dbl>                     <MULTIPOLYGON [°]>
## 1 E060000~ Portsmouth   -1.07  50.8 (((-1.071333 50.8364, -1.061647 50.83563, -~
## 2 E060000~ Southampton  -1.40  50.9 (((-1.401357 50.95039, -1.400148 50.94976, ~
## 3 E060000~ Isle of Wig~ -1.33  50.7 (((-1.301876 50.76673, -1.301853 50.76673, ~
## 4 E070000~ Fareham      -1.24  50.9 (((-1.269333 50.89862, -1.269281 50.89825, ~
## 5 E070000~ Gosport      -1.17  50.8 (((-1.113141 50.79134, -1.111902 50.78956, ~
## 6 E070000~ Basingstoke~ -1.22  51.3 (((-0.9861238 51.36285, -0.9854827 51.36203~
```

Which areas have we got?

```r
table(la_sf_data$lad18nm)
```

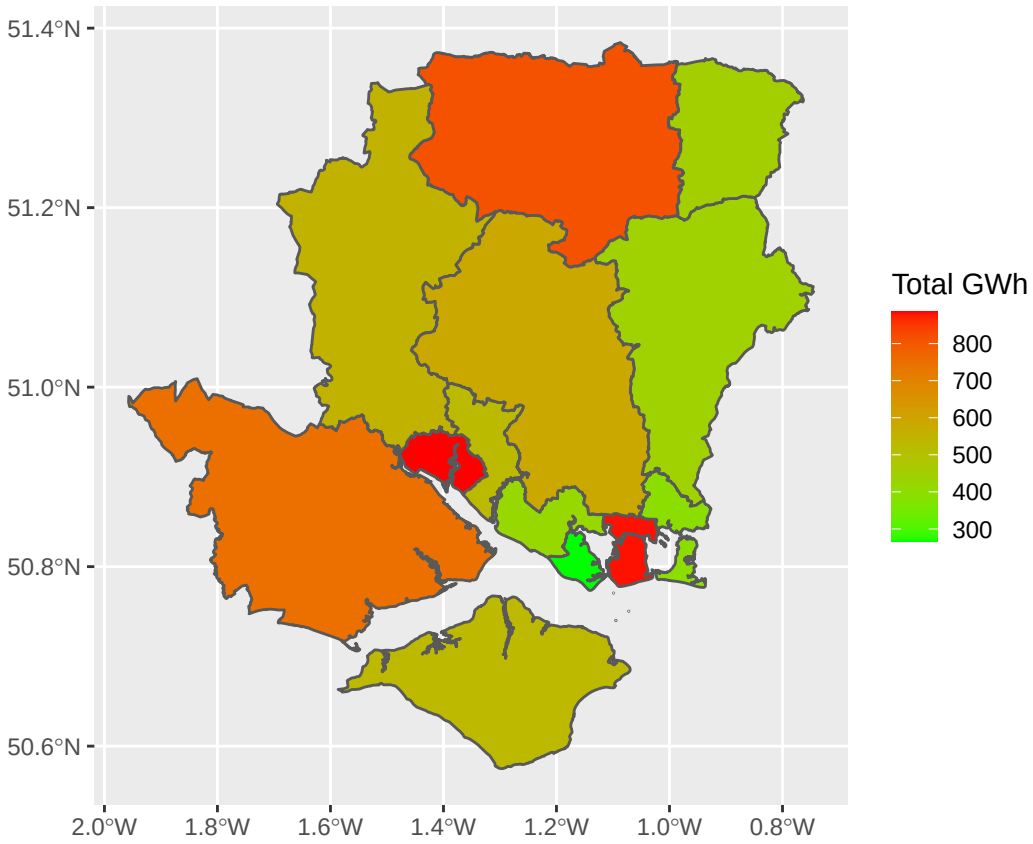
```
##
## Basingstoke and Deane        East Hampshire             Eastleigh
##                     1                     1                     1
##               Fareham               Gosport                  Hart
##                     1                     1                     1
##                Havant         Isle of Wight            New Forest
##                     1                     1                     1
##            Portsmouth           Southampton           Test Valley
##                     1                     1                     1
##            Winchester
##                     1
```

4.2 Mapping data

Now we'll map/plot some of the data using the `ggplot2` approach. To do that we need to merge the boundaries and the energy data so that we can `fill` the boundaries with a colour according to one of the variables.

```r
# create a variable with the LA code and the same name as in the sf_data
la_elecData$lad18cd <- la_elecData$`LA Code`
# merge them
la_merged_sf <- merge(la_sf_data, la_elecData)

# plot
ggplot2::ggplot(la_merged_sf) +
  geom_sf(aes(fill = Total_GWh)) +
  scale_fill_continuous(name = "Total GWh", low = "green", high = "red")
```



5 MSOAs

These are census areas - smaller than LSOAs so the files are bigger and the maps are denser.

5.1 Loading data

As before we're going to load some polygon boundary data (you need internet access for this) and some energy demand data so we can link them and map them.

```
# electricity consumption data at MSOA level (pre downloaded)
inFile <- here::here("data", "energy", "MSOA_Dom_Elec", "MSOA_DOM_ELEC_2019.csv")
msoa_elecData <- readr::read_csv(inFile)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   `Local Authority Name` = col_character(),
##   `Local Authority Code` = col_character(),
##   `MSOA Name` = col_character(),
##   `Middle Layer Super Output Area (MSOA) Code` = col_character(),
##   `Number of meters` = col_double(),
##   `Consumption (kWh)` = col_double(),
##   `Mean consumption (kWh per meter)` = col_double(),
##   `Median consumption (kWh per meter)` = col_double()
```

```
## )
```

This is electricity consumption data for 2019 for MSOAs. What variables have we got?

```
head(msoa_elecData)
```

```
## # A tibble: 6 x 8
##   `Local Authority N~ `Local Authority ~ `MSOA Name`  `Middle Layer Super Outpu~
##   <chr>               <chr>              <chr>        <chr>
## 1 Hartlepool          E06000001          Hartlepool ~ E02002483
## 2 Hartlepool          E06000001          Hartlepool ~ E02002484
## 3 Hartlepool          E06000001          Hartlepool ~ E02002485
## 4 Hartlepool          E06000001          Hartlepool ~ E02002487
## 5 Hartlepool          E06000001          Hartlepool ~ E02002488
## 6 Hartlepool          E06000001          Hartlepool ~ E02002489
## # ... with 4 more variables: Number of meters <dbl>, Consumption (kWh) <dbl>,
## #   Mean consumption (kWh per meter) <dbl>,
## #   Median consumption (kWh per meter) <dbl>
```

Clearly we are going to need to filter out the ones we don;t want...

Now load the MSOA boundary data for the Solent region only to keep things small & easy to play with.

We pre-downloaded these.

```
# las_to_load <- c("Southampton","Portsmouth","Winchester",
#                  "Eastleigh","Isle of Wight","Fareham",
#                  "Gosport","Test Valley","East Hampshire",
#                  "Havant","New Forest","Hart","Basingstoke and Deane")

# we pre-saved this data in the data folder of the repo for speed
# see getBoundaryData.R for how it works

inf <- here::here("data", "boundaries", "msoa_solent.shp") # use here to specify the data location
message("Loading MSOA geometry from ONS Open Geography API")
```

```
## Loading MSOA geometry from ONS Open Geography API
```

```
msoa_sf_data <- sf::read_sf(inf)

head(msoa_sf_data)
```

```
## Simple feature collection with 6 features and 13 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 458985.4 ymin: 102894 xmax: 467725.4 ymax: 107035
## Projected CRS: OSGB 1936 / British National Grid
## # A tibble: 6 x 14
##   MSOA11CD  MSOA11NM    OBJECTID  BNG_E  BNG_N LONG_   LAT Shape_Leng Shape__Are
##   <chr>     <chr>          <int>  <int>  <int> <dbl> <dbl>      <dbl>      <dbl>
## 1 E02003524 Portsmouth~     3510 463726 106331 -1.10  50.9      8175.   1880016.
## 2 E02003525 Portsmouth~     3511 465172 105904 -1.08  50.8     13220.   2012966.
```

```
## 3 E02003526 Portsmouth~     3512 466581 106095 -1.06  50.9     10425.   2260228.
## 4 E02003527 Portsmouth~     3513 464176 104420 -1.09  50.8     21957.   3838768.
## 5 E02003529 Portsmouth~     3514 466420 104925 -1.06  50.8     10636.   1499618.
## 6 E02003530 Portsmouth~     3515 465411 103721 -1.07  50.8     13264.   1695461.
## # ... with 5 more variables: Shape__Len <dbl>, LAD11CD <chr>, LAD11NM <chr>,
## #   nOAs <int>, geometry <MULTIPOLYGON [m]>
```

Which areas have we got and how many MSOAs are there in each?

```
table(msoa_sf_data$LAD11NM)
```

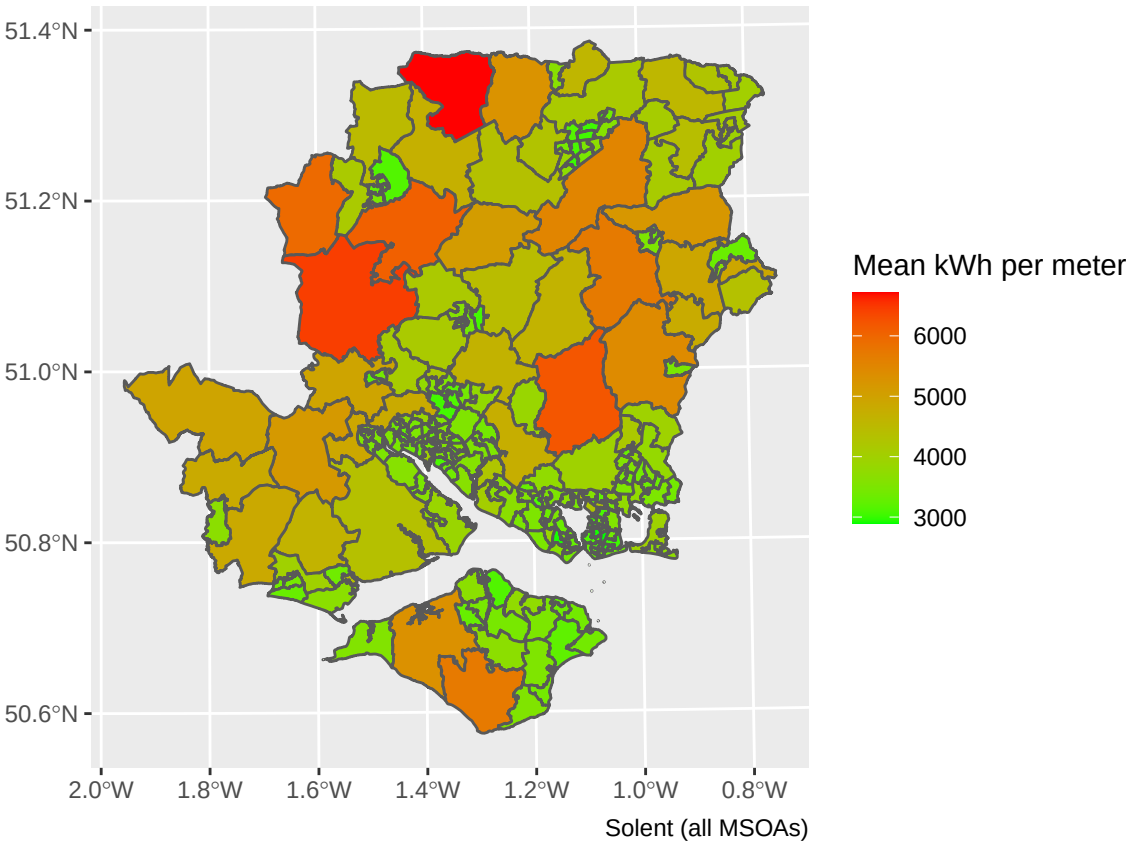
```
##
## Basingstoke and Deane        East Hampshire             Eastleigh
##                    22                    15                    15
##               Fareham               Gosport                  Hart
##                    14                    10                    11
##                Havant         Isle of Wight           New Forest
##                    17                    18                    23
##            Portsmouth           Southampton          Test Valley
##                    25                    32                    15
##            Winchester
##                    14
```

5.2 Mapping data

Now we'll map/plot some of the data using the `ggplot2` approach. To do that we need to merge the boundaries and the energy data so that we can `fill` the boundaries with a colour according to one of the variables.

```
# create a variable with the LA code and the same name as in the sf_data
msoa_elecData$MSOA11CD <- msoa_elecData$`Middle Layer Super Output Area (MSOA) Code`
# merge them
msoa_merged_sf <- merge(msoa_sf_data, msoa_elecData)

# plot
ggplot2::ggplot(msoa_merged_sf) +
  geom_sf(aes(fill = `Mean consumption (kWh per meter)`)) +
  scale_fill_continuous(name = "Mean kWh per meter", low = "green", high = "red") +
  labs(caption = "Solent (all MSOAs)")
```



6 LSOAs

Check this works with BEIS LSOA level electricity data.

```
inFile <- here::here("data", "energy", "LSOA_Dom_Elec", "LSOA_ELEC_2019.csv")
lsoa_elecData <- readr::read_csv(inFile)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   `Local Authority Name` = col_character(),
##   `Local Authority Code` = col_character(),
##   `Middle Layer Super Output Area (MSOA) Name` = col_character(),
##   `Middle Layer Super Output Area (MSOA) Code` = col_character(),
##   `Lower Layer Super Output Area (LSOA) Name` = col_character(),
##   `Lower Layer Super Output Area (LSOA) Code` = col_character(),
##   `Total number of domestic electricity meters` = col_double(),
##   `Total domestic electricity consumption (kWh)` = col_double(),
##   `Mean domestic electricity consumption
## (kWh per meter)` = col_double(),
##   `Median domestic electricity consumption
## (kWh per meter)` = col_double()
## )
```

```
lsoa_elecData$LSOA11CD <- lsoa_elecData$`Lower Layer Super Output Area (LSOA) Code`
```

```
# las_to_load <- c("Southampton","Portsmouth","Winchester",
#                  "Eastleigh","Isle of Wight","Fareham",
#                  "Gosport","Test Valley","East Hampshire",
#                  "Havant","New Forest","Hart","Basingstoke and Deane")

# we pre-saved this data in the data folder of the repo for speed
# sourced from: https://geoportal.statistics.gov.uk/search?collection=Dataset&sort=name&tags=all(BDY_LS

# see getBoundaryData.R for how it works

inf <- here::here("data", "boundaries", "lsoa_solent.shp") # use here to specify the data location
message("Loading LSOA geometry from file")
```

```
## Loading LSOA geometry from file
```

```
lsoa_sf_data <- sf::read_sf(inf)


head(lsoa_sf_data)
```

```
## Simple feature collection with 6 features and 15 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 465991.4 ymin: 100849 xmax: 467770.5 ymax: 102360.8
## Projected CRS: OSGB 1936 / British National Grid
## # A tibble: 6 x 16
##   LSOA11CD  OBJECTID LSOA11NM   LSOA11NMW    BNG_E  BNG_N  LONG   LAT Shape__Are
##   <chr>        <int> <chr>      <chr>        <int>  <int> <dbl> <dbl>      <dbl>
## 1 E01017013    16519 Portsmout~ Portsmouth~ 466127 101741 -1.06  50.8    145134.
## 2 E01017014    16520 Portsmout~ Portsmouth~ 466924 101957 -1.05  50.8    757371.
## 3 E01017015    16521 Portsmout~ Portsmouth~ 467084 101499 -1.05  50.8    472647.
## 4 E01017016    16522 Portsmout~ Portsmouth~ 466330 101597 -1.06  50.8    122927.
## 5 E01017017    16523 Portsmout~ Portsmouth~ 466813 101069 -1.05  50.8    166924.
## 6 E01017018    16524 Portsmout~ Portsmouth~ 466350 101184 -1.06  50.8    184945.
## # ... with 7 more variables: Shape__Len <dbl>, MSOA11CD <chr>, MSOA11NM <chr>,
## #   LAD11CD <chr>, LAD11NM <chr>, nLSOAs <int>, geometry <MULTIPOLYGON [m]>
```

Which areas have we got and how many MSOAs are there in each?

```
table(lsoa_sf_data$LAD11NM)
```

```
##
## Basingstoke and Deane        East Hampshire             Eastleigh
##                   109                    72                    77
##               Fareham               Gosport                  Hart
##                    73                    53                    57
##                Havant         Isle of Wight           New Forest
##                    78                    89                   114
##           Portsmouth           Southampton          Test Valley
```

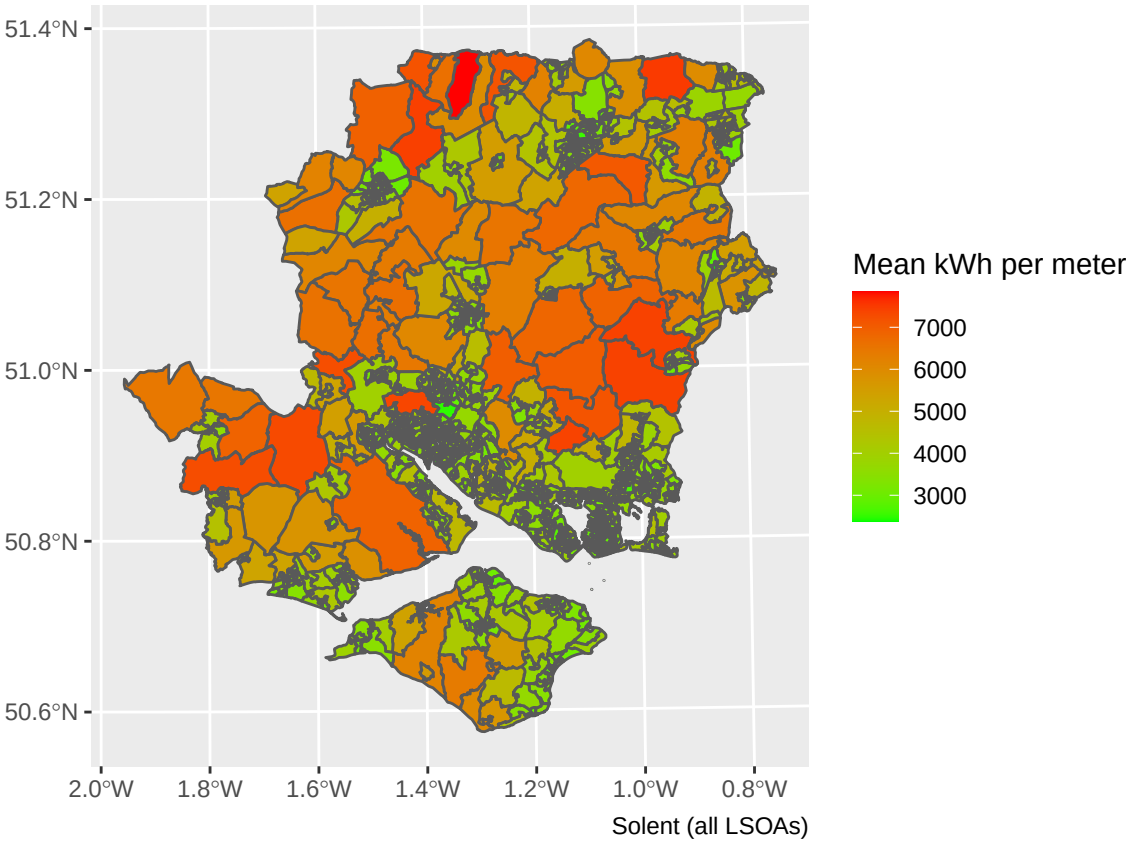
```
##                 125                 148                  71
##          Winchester
##                  70
```

6.1 Mapping data

Now we'll map/plot some of the data using the `ggplot2` approach. To do that we need to merge the boundaries and the energy data so that we can `fill` the boundaries with a colour according to one of the variables.

```
lsoa_merged_sf <- merge(lsoa_sf_data, lsoa_elecData)

# plot
ggplot2::ggplot(lsoa_merged_sf) +
  geom_sf(aes(fill = `Mean domestic electricity consumption \n(kWh per meter)`)) +
  scale_fill_continuous(name = "Mean kWh per meter", low = "green", high = "red") +
  labs(caption = "Solent (all LSOAs)")
```



Cities disappear due to the density of boundaries. As an example, this is the map for just Southampton.

```
library(dplyr) # for filter
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:raster':
```

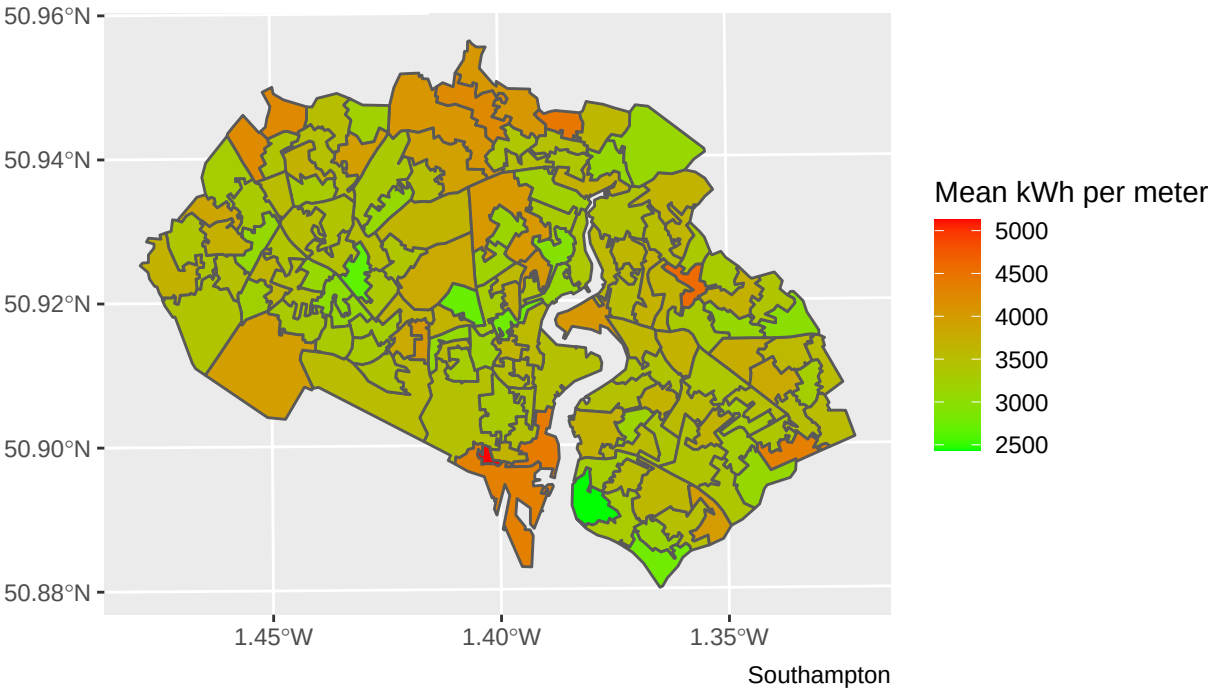
```
##
##     intersect, select, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
mapData <- dplyr::filter(lsoa_merged_sf, LAD11NM == "Southampton")

# plot
ggplot2::ggplot(mapData) +
  geom_sf(aes(fill = `Mean domestic electricity consumption \n(kWh per meter)`)) +
  scale_fill_continuous(name = "Mean kWh per meter", low = "green", high = "red") +
  labs(caption = "Southampton")
```



#

References