

5. Basic Maps

This helpsheet shows you how to make a simple map using the `GISTools` package.

To start with, we need to load the `GISTools` package as well as some other packages we need:

```
library(rgdal)
library(GISTools)
library(RColorBrewer)
```

We also need to load a data set, which in this example, relate to Lower Layer Super Output (LSOA) zones within Liverpool, and also an outline of England. The following commands will set your working directory, download, unzip and load the data files.

```
# Set working directory
setwd("M:/R work")
# Download data.zip from the web
download.file("http://data.alex-singleton.com/r-helpsheets/5/data.zip", "data.zip")
# Unzip file
unzip("data.zip")

# Read in both shapefiles
LSOA <- readOGR(".", "england_LSOA_2011_dwelling_count")
outline <- readOGR(".", "England_ol_2011_gen_clipped")
```

We can do a very basic plot of the map using:

```
plot(LSOA)
```

Which gives us a map, just showing the boundaries of the LSOAs.

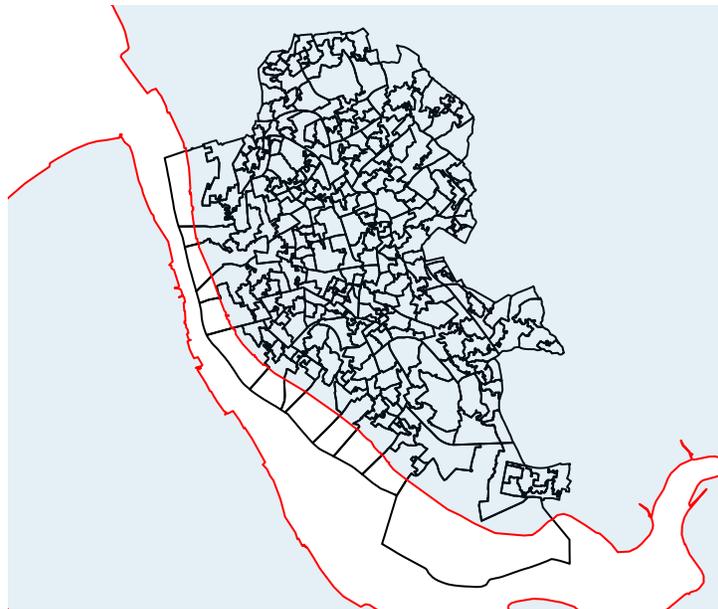


We can also plot an outline of England in a similar way.

```
plot(outline)
```

This replaces the first map, but we can get R to overlay one on top of the other, by using the command `add = TRUE`. The order of plots is key here - R will maintain the scale and extent of the first map. We can also adjust the colour of the border to a red colour (`border="red"`), and the fill colour (`col="#2C7FB820"`) a shade of blue. These represent two ways of specifying colours. The second contains eight alphanumeric characters, the first six relate to a HEX colour code. To view various colours that can be used in R, have a look at the website <http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf>. The final two characters are the level of transparency (in this case 20%). *Sometimes when running R in Windows, the transparency option will not work - it will just fill it with a solid colour. In this case, just remove the `col = "#2C7FB820"` section from the plot command to just generate a red outline.*

```
# Plot the LSOA Map
plot(LSOA)
# Overplot the outline map
plot(outline, add = TRUE, border = "red", col = "#2C7FB820")
```



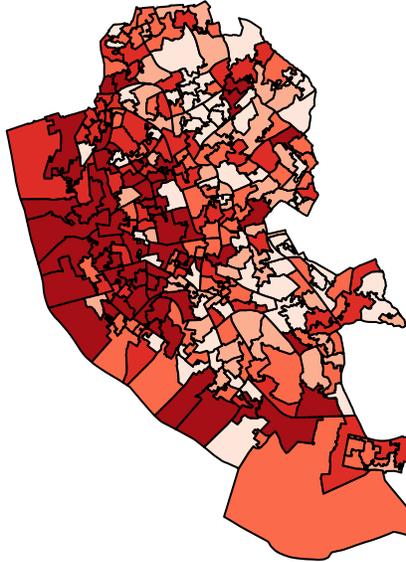
The LSOA data frame contains some more information, which we can see by looking in the data slot of the object:

```
head(LSOA@data)
```

	GID	ZONECODE	LSOA_NAME	COUNT_DWEL
0	458	E01006739	Liverpool 058B	719
1	4350	E01006687	Liverpool 056D	926
2	4351	E01006741	Liverpool 058C	776
3	4352	E01006743	Liverpool 058D	994
4	4370	E01006528	Liverpool 052C	646
5	4371	E01006684	Liverpool 056B	982

This shows us that the shape file contains a field called 'COUNT_DWELL' which contains the count of the number of dwellings in each LSOA. We can use this to create a choropleth map with:

```
choropleth(LSOA, LSOA$COUNT_DWEL)
```



This map is ok, but we can easily make it more effective with a few extra commands. The new commands include:

1. `brewer.pal` which returns a set of colours from a range of pre-set palettes that look good on maps. In this case, we are getting 5 colours from the "Blues" palette. For more information on the R command, type `?brewer.pal` into R, for more information on the concept, see <http://colorbrewer.org>.
2. `auto.shading` which categorises the data we want to show on to the map (in this case, `LSOA$COUNT_DWELL`) into the specified number of categories (5), coloured with the specified colours (`cols = brewer.pal(5, "Blues")`).
3. `choro.legend` and `north.arrow` both have a set of coordinates as one of their parameters (e.g. 331089, 384493). These say where the object is located on the map. You may have to fiddle with these to get the spacing correct (see note below).

Run the commands below in R, and read the text below for more information.

```
# Set colour and number of classes
shades <- auto.shading(LSOA$COUNT_DWEL, n = 5, cols = brewer.pal(5, "Blues"))
# Draw the map
choropleth(LSOA, LSOA$COUNT_DWEL, shades)
# Add a legend
choro.legend(331089, 384493, shades, fmt = "%g", title = "Count of Dwellings")
# Add a title to the map
title("Count of Dwellings by LSOA, 2011")
# add North arrow
north.arrow(332308, 387467, 300)
# Draw a box around the map
box(which = "outer")
```

See the next page for the map.

You might find you will need to adjust the location or size of the legend to get this to fit onto your plot correctly. To find a new set of location coordinates, type `locator()` into the terminal and press enter. After doing this, when you hover over the plot, the mouse will turn into a cross. If you click, and then right-click

and choose 'Stop', the location of the click is printed to the terminal - you can use these to re-position items in the plot.

To change the size of the legend, use the `cex =` command. Update the `choro.legend` line to read `choro.legend(328089, 384493, shades, fmt = "%g", title = "Count of Dwellings", cex = 1.1)` and see what happens. The `cex` value is a multiple which increases or decreases the size of the legend. Experiment with this until you find something that works well.

For more information on the `GISTools` package, have a look at <http://cran.r-project.org/web/packages/GISTools/GISTools.pdf>.

