Practical 6: Mapping Point Data in R

An Introduction to Spatial Data Analysis and Visualisation in R - Guy Lansley & James Cheshire (2016)

This practical will follow on from the previous exercise (https://data.cdrc.ac.uk/tutorial/aa5491c9-cbac-4026-97c9-f9168462f4ac/70c4bc61-0475-4806-9240-4ef1fa649a06) by introducing the handling and mapping of spatial point data in R using tmap. Data for the practical can be downloaded from the **Introduction to Spatial Data Analysis and Visualisation in R** (https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r) homepage.

In this tutorial we will:

- Create a point shapefile from a CSV using coordinates
- Map the points in tmap
- Create a proportional bubble map

First, we must set the working directory and load the practical data.

```
# Set the working directory
setwd("C:/Users/Guy/Documents/Teaching/CDRC/Practicals")

# Load the data. You may need to alter the file directory
Census.Data <-read.csv("practical_data.csv")
```

We will also need the polygon shapefile from our previous exercise and to join our census data to it.

```
# load the spatial libraries
library("rgdal")
library("rgeos")

# Load the output area shapefiles
Output.Areas <- readOGR(".", "Camden_oa11")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "Camden_oa11"
## with 749 features
## It has 1 fields
```

```
# join our census data to the shapefile
OA.Census <- merge(Output.Areas, Census.Data, by.x="OA11CD", by.y="OA")
```

Loading point data into R

In this tutorial we will be handling house price paid data originally made available for free by the Land Registry. The data is formatted as CSV where each row is a unique house sale, including the price paid in pounds and the postcode. Prior to this practical, the data file was joined to the Office for National Statistics (ONS) postcode lookup table which provides latitude and longitude coordinates for each postcode.

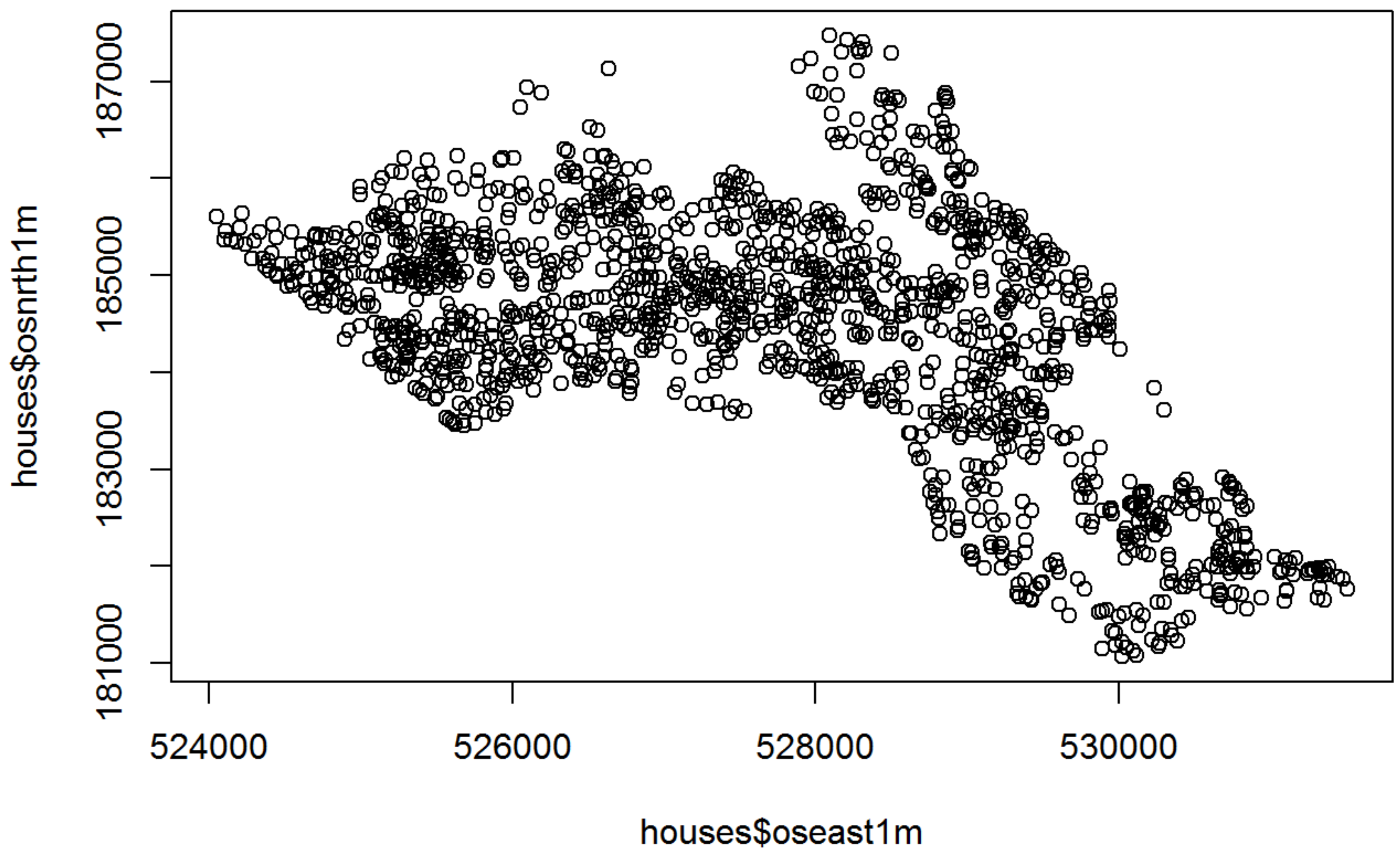
```r
# load the house prices csv file
houses <- read.csv("CamdenHouseSales15.csv")

# we only need a few columns for this practical

houses <- houses[,c(1,2,8,9)]
```

Whilst it is possible to plot this data using the standard `plot()` in R (as demonstrated below), it is not being handled as spatial data.

```r
# 2D scatter plot
plot(houses$oseast1m, houses$osnrth1m)
```



Therefore, we need to assign spatial attributes to the CSV so it can be mapped properly in R. To do this we will need to load the sp package, this package provides classes and methods for handling spatial data. Remember to install the package first if you have not done so before.

Next, we will convert the CSV into a SpatialPointsDataFrame. To do this we will need to set what the data is to be included, what columns contain the x and y coordinates, and what projection system we are using.

```r
library("sp")

# create a House.Points SpatialPointsDataFrame
House.Points <-SpatialPointsDataFrame(houses[,3:4], houses, proj4string = CRS("+in
it=EPSG:27700"))
```

Before we map the points, we will create a base map using the output area boundaries.

```
library("tmap")

# This plots a blank base map, we have set the transparency of the borders to 0.4
tm_shape(OA.Census) + tm_borders(alpha=.4)
```

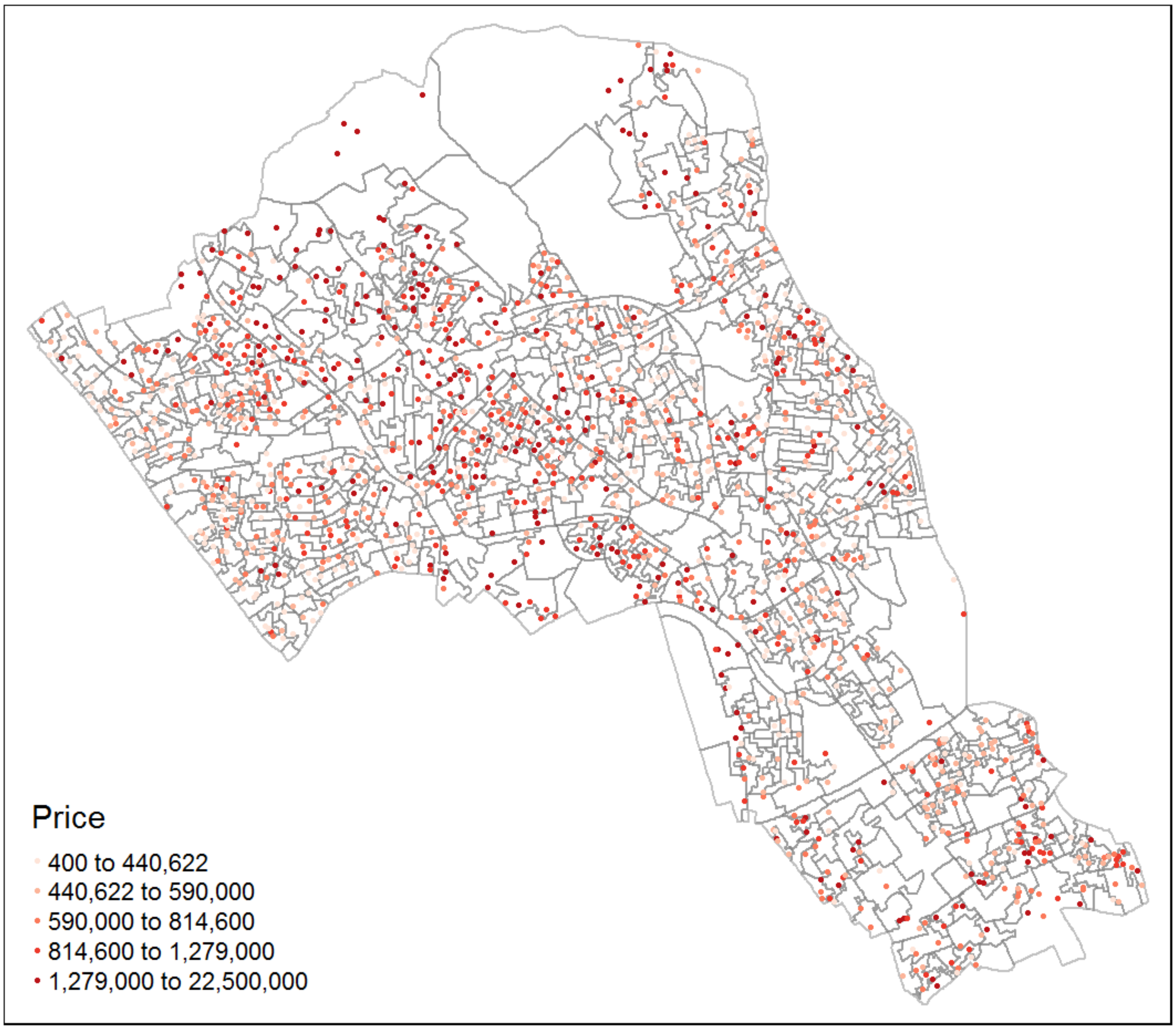
We can now add on the points as an additional *tm_shape* layer in our map.

To do this, we copy in the same code to make our base map, followed by a plus symbol, then enter the details for the points data. The additional arguments for the points data can be summarised as:

```
tm_shape(polygon file) + tm_borders(transparency = 40%) +
tm_shape(our spatial points data frame) + tm_dots(what variable is coloured, the c
olour palette and interval style)
```

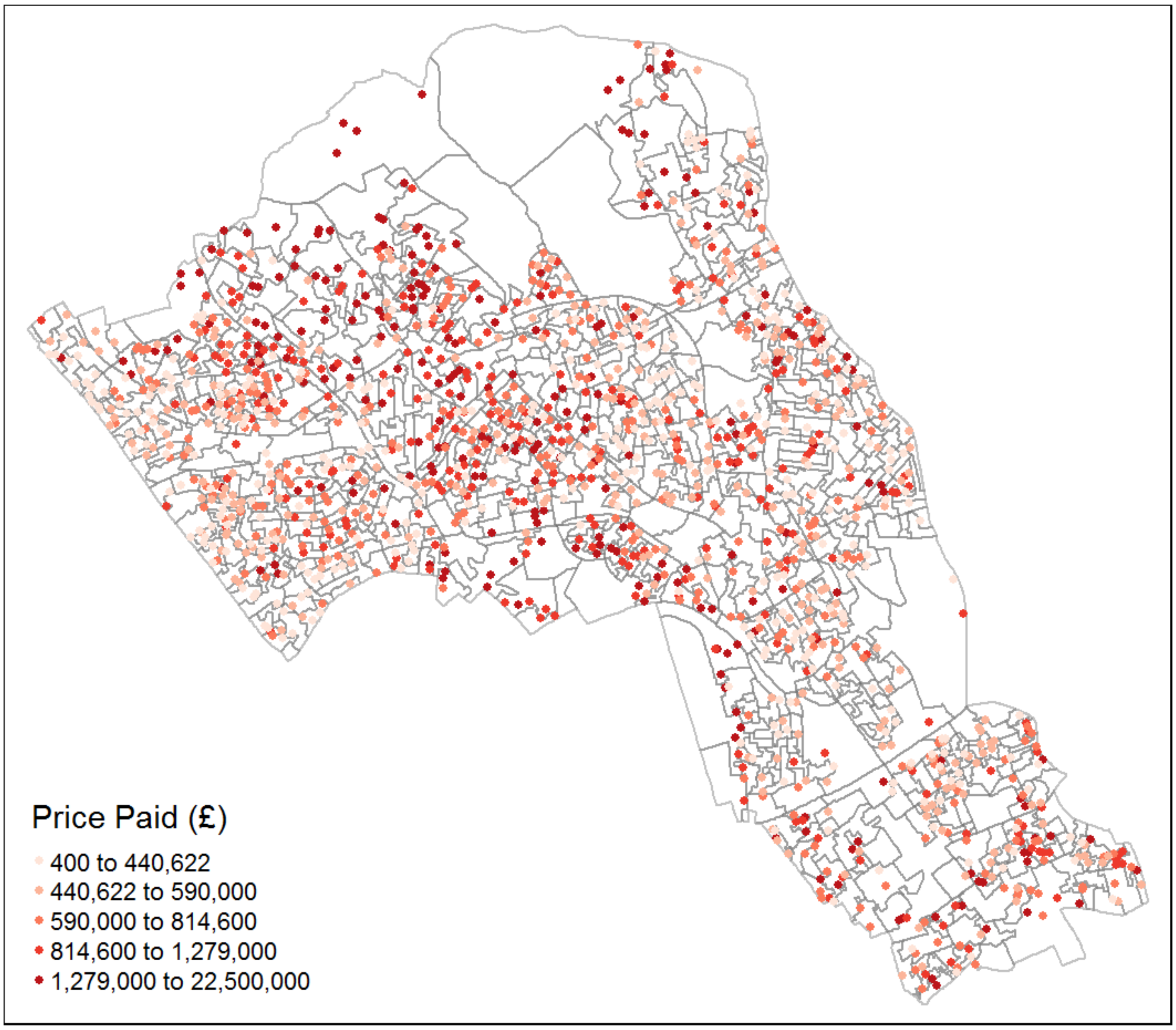
Which is entered into R like this:

```
# creates a coloured dot map
tm_shape(OA.Census) + tm_borders(alpha=.4) +
tm_shape(House.Points) + tm_dots(col = "Price", palette = "Reds", style = "quantil
e")
```



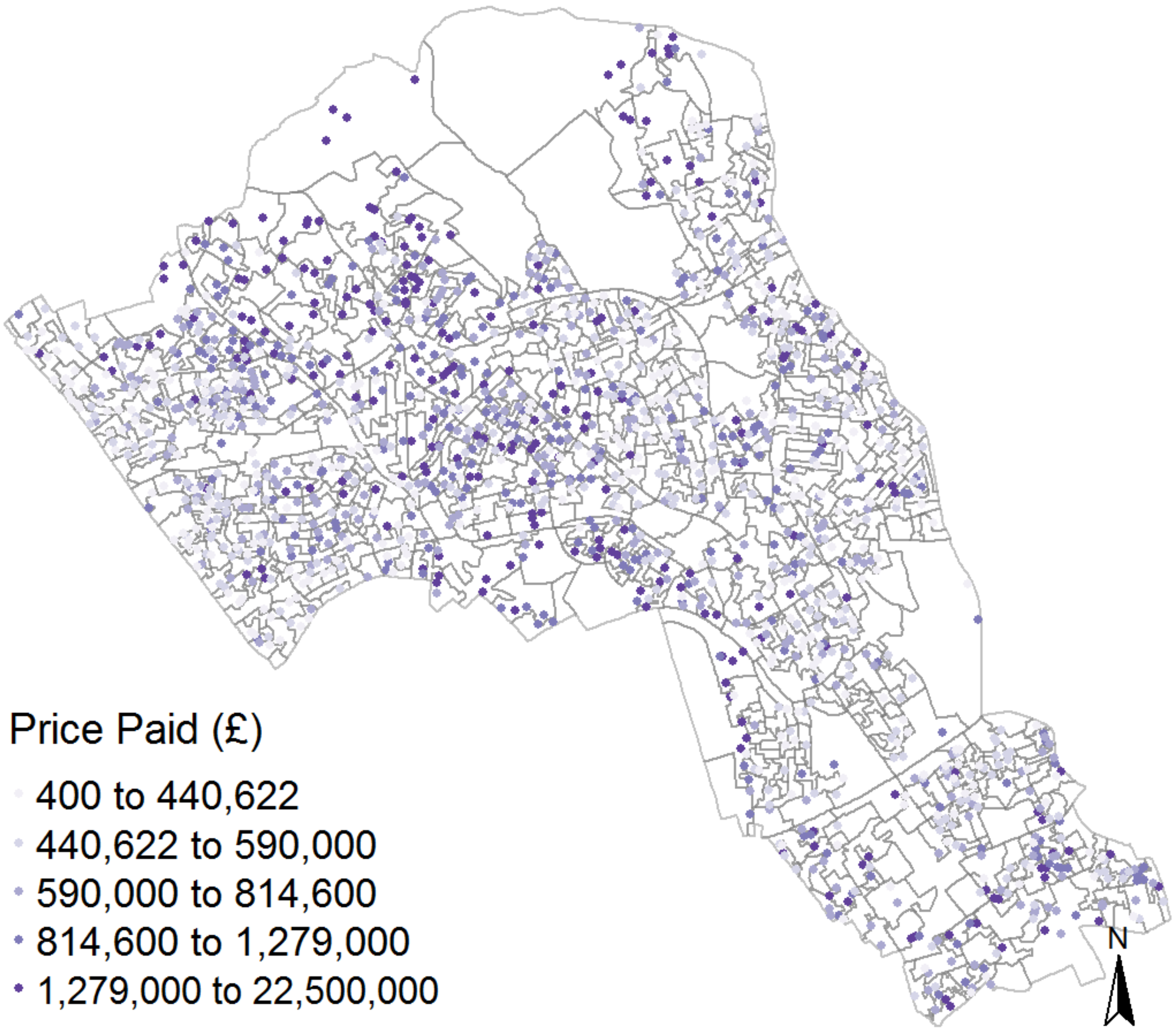
We can also add in more arguments within the `tm_dots()` function for points like we would with `tm_fill()` for polygon data. Some arguments are unique to `tm_dots()`. For example, the `scale` argument which rescales the size of the points.

```
# creates a coloured dot map
tm_shape(OA.Census) + tm_borders(alpha=.4) +
tm_shape(House.Points) + tm_dots(col = "Price", scale = 1.5, palette = "Reds", sty
le = "quantile", title = "Price Paid (£)")
```



We can also add `tm_layout()` and `tm_compass()` as we did in the previous practical.

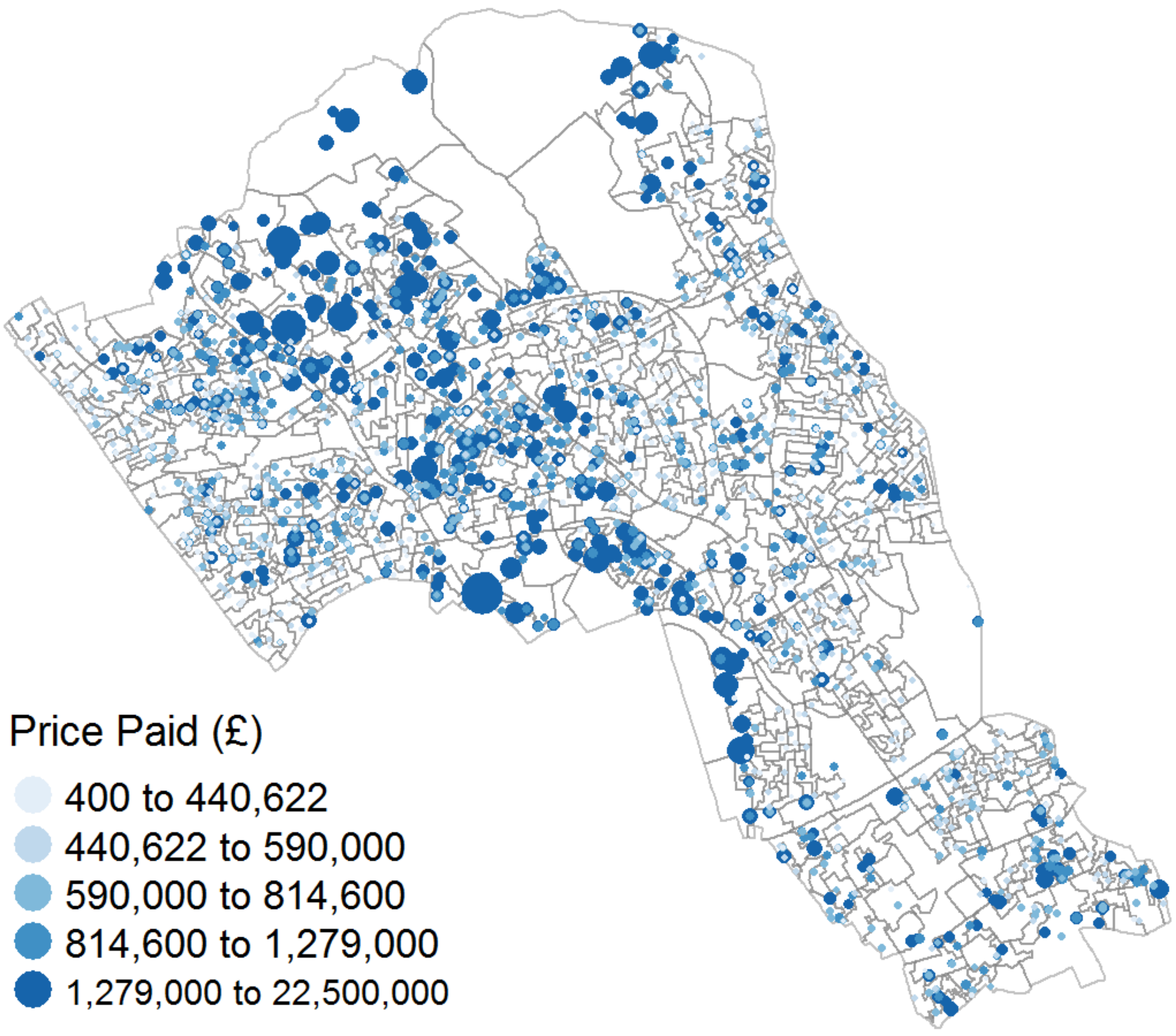
```
# creates a coloured dot map
tm_shape(OA.Census) + tm_borders(alpha=.4) +
tm_shape(House.Points) + tm_dots(col = "Price", scale = 1.5, palette = "Purples",
style = "quantile", title = "Price Paid (£)")  +
tm_compass() +
tm_layout(legend.text.size = 1.1, legend.title.size = 1.4, frame = FALSE)
```



Proportional symbols

We can also create proportional symbols in tmap. To do it in tmap, we replace the `tm_dots()` function with the `tm_bubbles()` function. In the example below, the size and colours are both set as the price column.

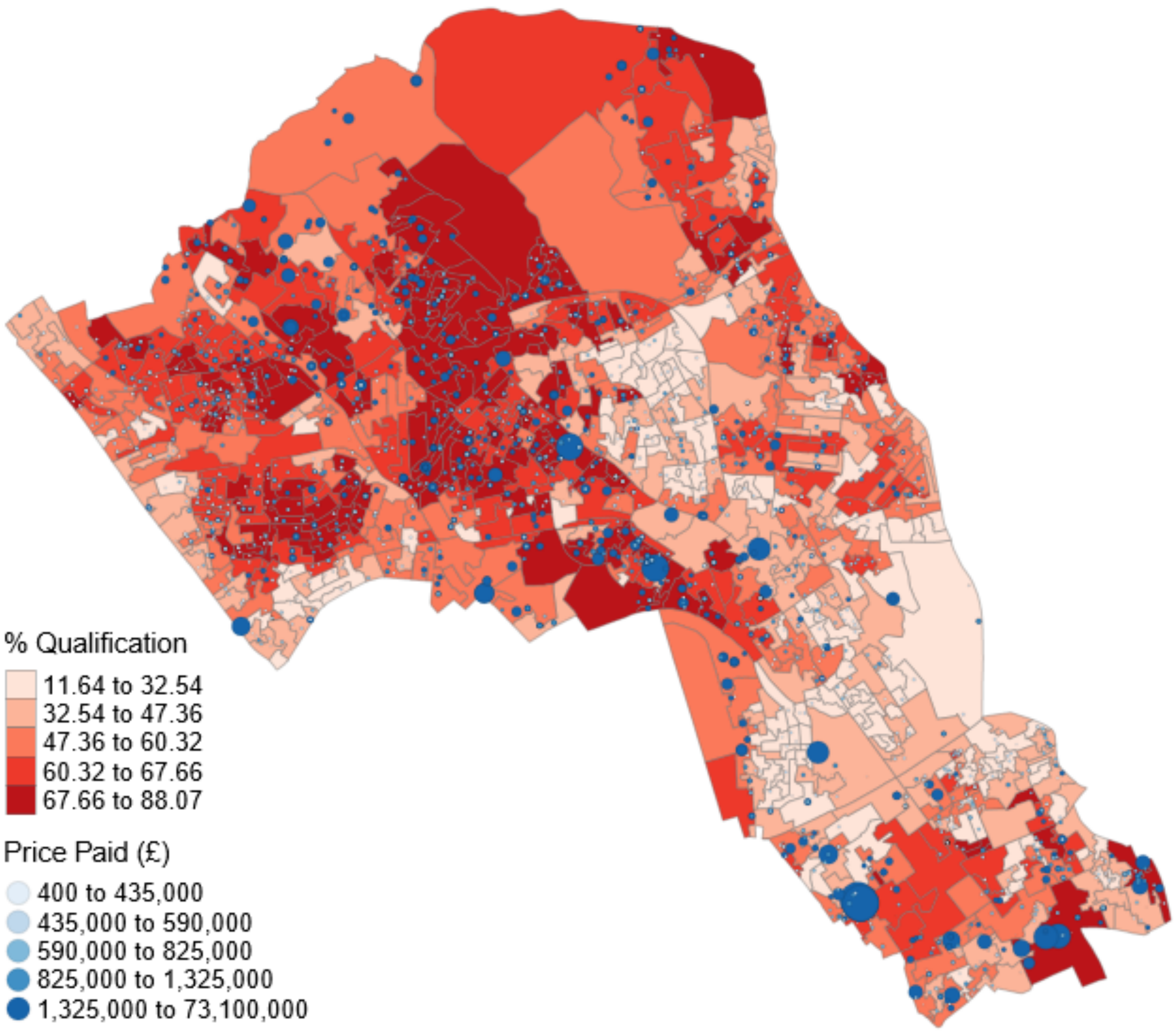
```
# creates a proportional symbol map
tm_shape(OA.Census) + tm_borders(alpha=.4) +
tm_shape(House.Points) + tm_bubbles(size = "Price", col = "Price", palette = "Blue
s", style = "quantile", legend.size.show = FALSE, title.col = "Price Paid (£)") +
tm_layout(legend.text.size = 1.1, legend.title.size = 1.4, frame = FALSE)
```



We can also make the polygon shapefile display one of our census variables as a choropleth map as shown below. In this example, we have also added some more parameters within the tm_bubbles() function to create thin borders around the bubbles.

```
# creates a proportional symbol map
tm_shape(OA.Census) + tm_fill("Qualification", palette = "Reds", style = "quantile
", title = "% Qualification") +
tm_borders(alpha=.4) +
tm_shape(House.Points) + tm_bubbles(size = "Price", col = "Price", palette = "Blue
s", style = "quantile", legend.size.show = FALSE, title.col = "Price Paid (£)", bo
rder.col = "black", border.lwd = 0.1, border.alpha = 0.1) +
tm_layout(legend.text.size = 0.8, legend.title.size = 1.1, frame = FALSE)
```

This is an exported copy of the map …



Saving the shapefile

Finally, we can write the newly formed House.Points shapefile to our working directory if you wish to save it for later use.

```
# write the shapefile to your computer (remember to chang the dsn to your workspac
e)
writeOGR(House.Points, dsn = "C:/Users/Guy/Documents/Teaching/CDRC/Practicals", la
yer =  "Camden_house_sales", driver="ESRI Shapefile")
```

The rest of the online tutorials in this series can be found at: https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r (https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r)