# Practical 8: Representing Densities in R

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

This practical will introduce you to running a kernel density estimation in R. Kernel density estimation is a commonly used means of representing densities of spatial data points. The technique produces a smooth and continuous surface where each pixel represents a density value based on the number of points within a given distance bandwidth. We will also cover some basic techniques in handling and editing raster shapefiles. 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:

- Run a kernel density estimation
- Create a raster shapefile
- Map a raster shapefile in tmap
- Use a mask technique to clip a raster

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

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

We will also need to the load our spatial data files.

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

# Load the output area shapefiles, we won't join it to any data this time
Output.Areas <- readOGR(".", "Camden_oa11")
```

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

```
# load the houses point files
House.Points <- readOGR(".", "Camden_house_sales")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "Camden_house_sales"
## with 2547 features
## It has 4 fields
```

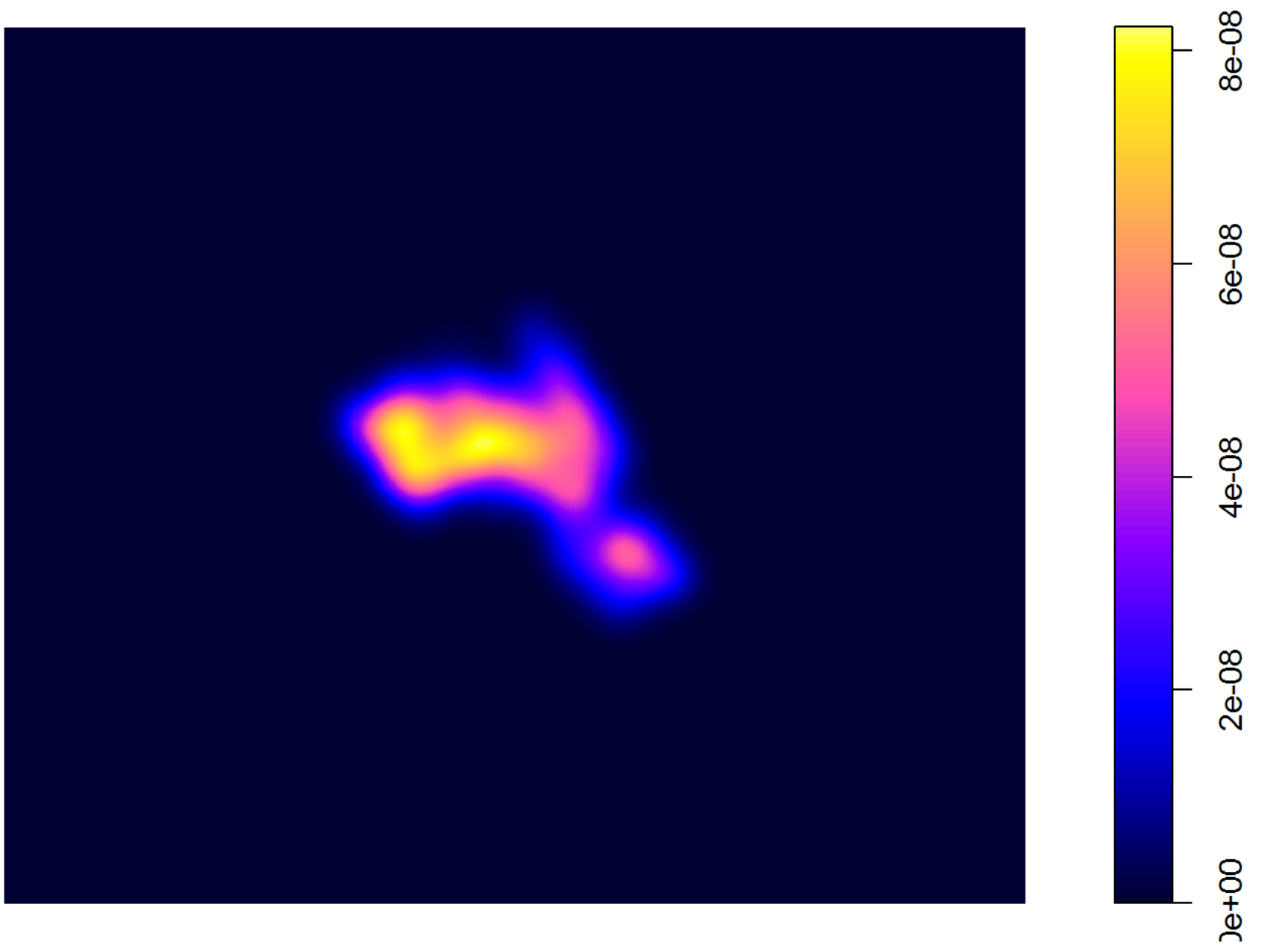
## Point densities

Whilst it is straight-forward to determine the frequency of phenomena across space with polygon data, it is more complicated to measure densities of points coherently. A relatively simple approach would be to use polygon zones to spatially aggregate the data (as we did in practical 7), and to then count the number of occurrences in each area.

However, techniques which have been devised to represent the densities of data points across two-dimensions also exist. One such approach, known as kernel density estimation, uses a moving quadrant to calculate the density for each area within a given threshold. For more information on the statistics behind kernel density measures please read the Geospatial Ananalysis (de Smith *et al*, 2015) (http://www.spatialanalysisonline.com/HTML/?building_blocks_of_spatial_ana.htm) webpage on *density, kernels and occupancy* (http://www.spatialanalysisonline.com/HTML/?building_blocks_of_spatial_ana.htm)

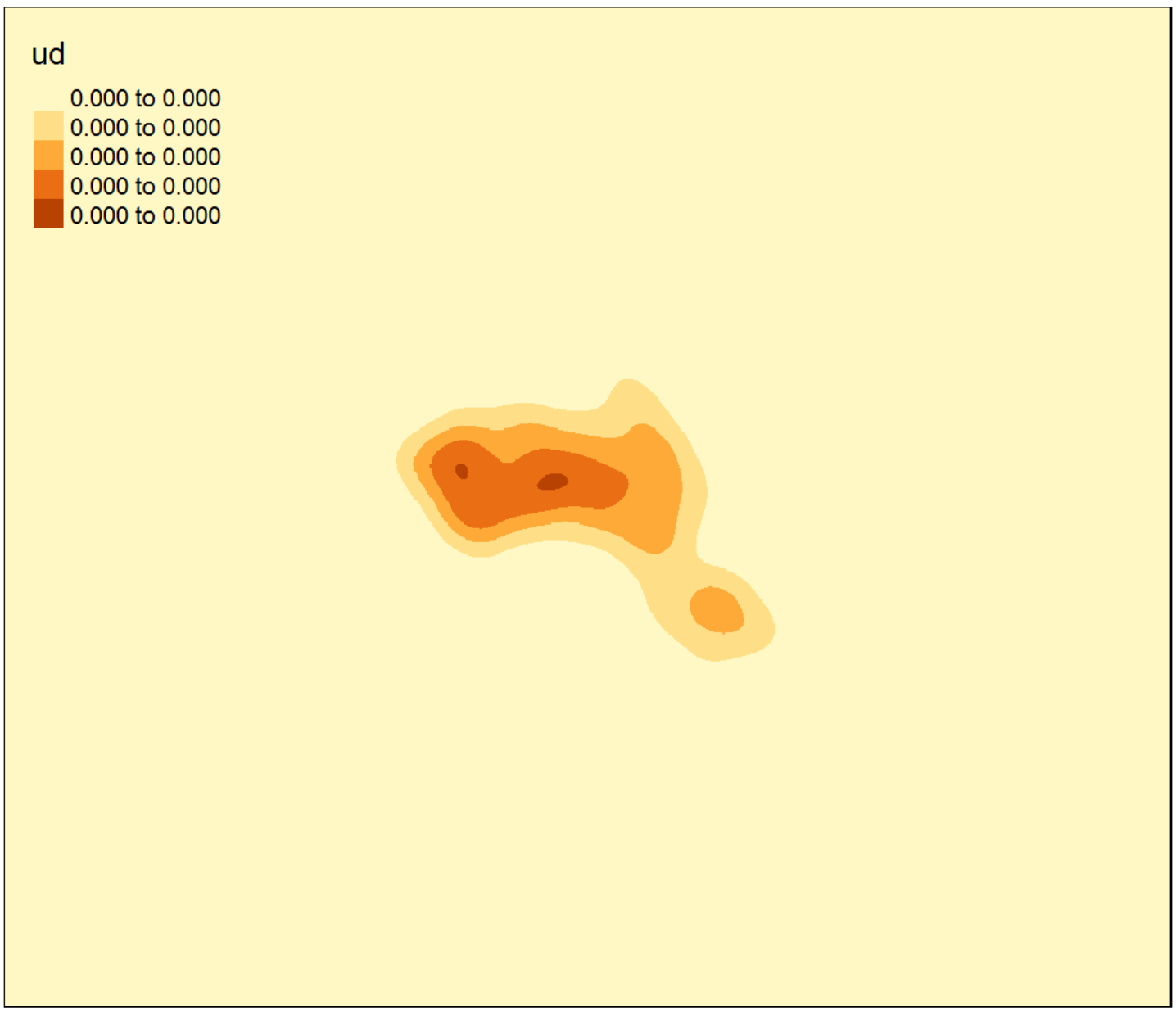
The following code will run a kernel density estimation for our Land Registry house price data. There are several different ways to run this through R, we will use the functions available from the *adehabitatHR* package.

```
# load the spatial libraries
library(raster)
library(adehabitatHR)

# runs the kernel density estimation,look up the function parameters for more opti
ons
kde.output <- kernelUD(House.Points, h="href", grid = 1000)

plot(kde.output)
```



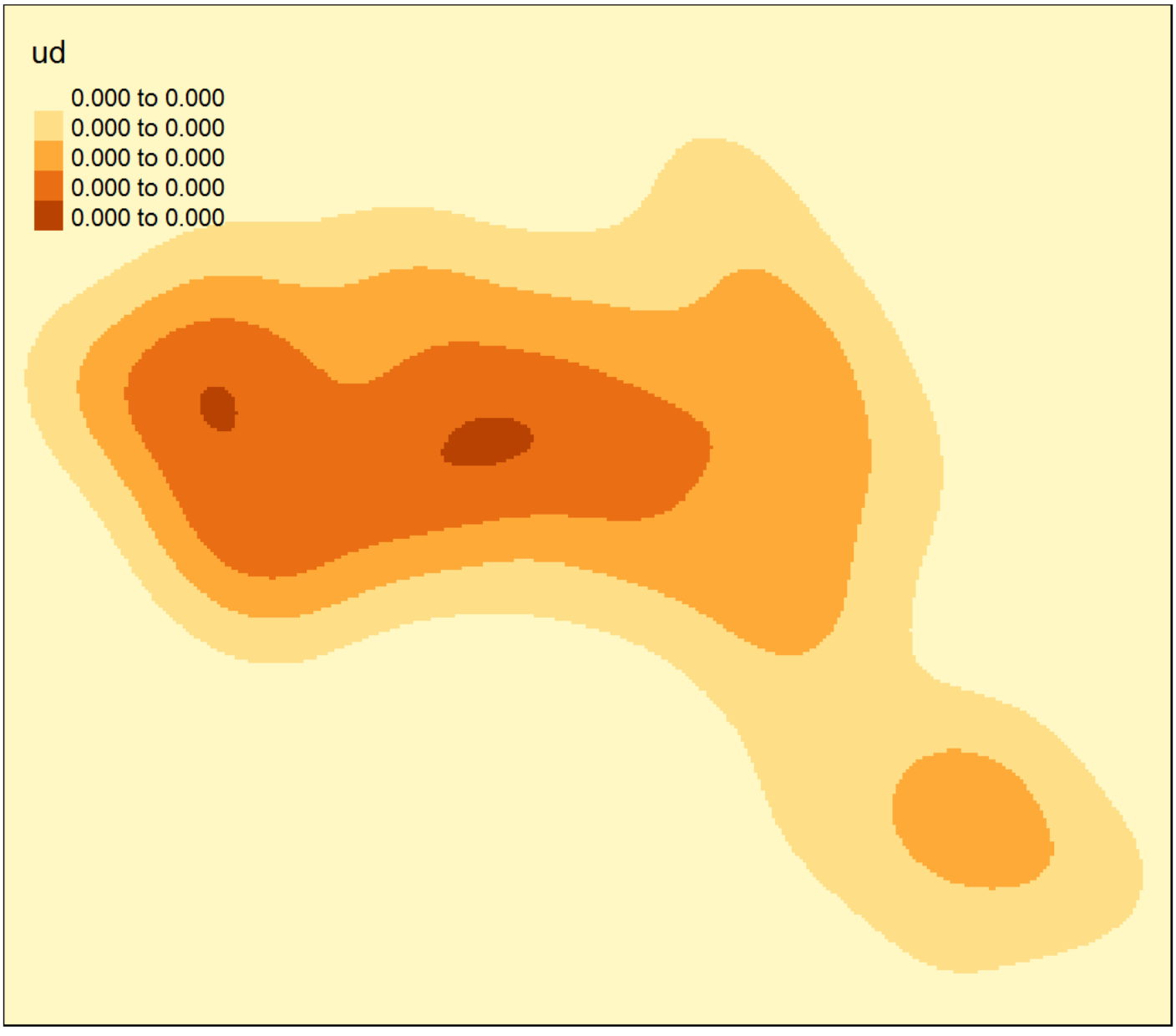
We can also create simple contour plots easily in R. Simply enter `contour(kde)` into R. However, to map the raster in tmap, we first need to ensure it has been projected correctly.

```
# converts to raster
kde <- raster(kde.output)
# sets projection to British National Grid
projection(kde) <- CRS("+init=EPSG:27700")

library(tmap)

# maps the raster in tmap, "ud" is the density variable
tm_shape(kde) + tm_raster("ud")
```



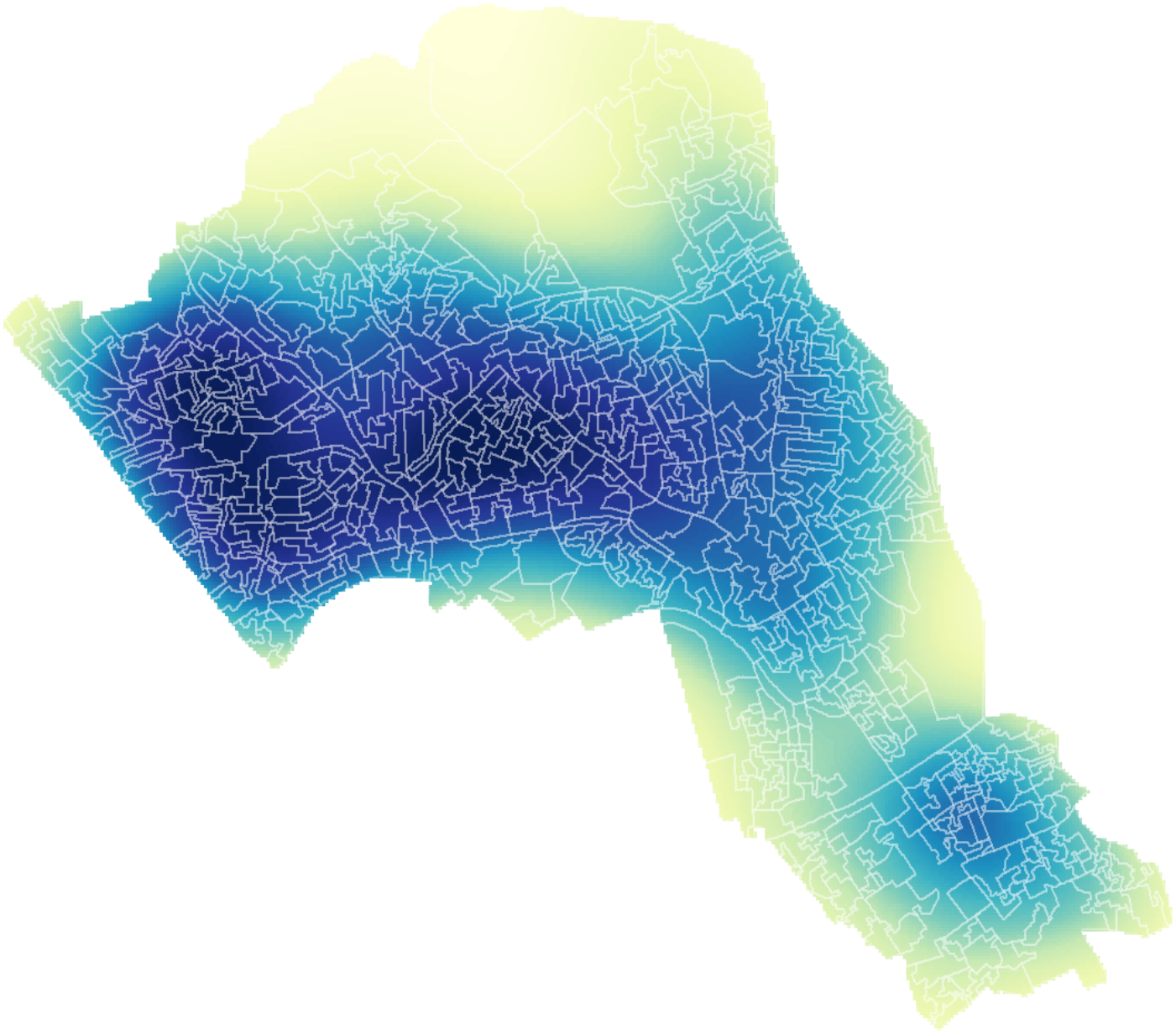

We can just about make out the shape of Camden. However, in this case, the raster includes a lot of empty space, we can zoom in on Camden by setting the map to the extents of a bounding box.

```
# creates a bounding box based on the extents of the Output.Areas polygon
bounding_box <- bb(Output.Areas)

# maps the raster within the bounding box
tm_shape(kde, bbox = bounding_box) + tm_raster("ud")
```




We can also mask (or clip) the raster by the output areas polygon and tidy up the graphic. This operation only preserves the parts of the raster which are within the spatial extent of the masking polygon.

```
# mask the raster by the output area polygon
masked_kde <- mask(kde, Output.Areas)

# maps the masked raster, also maps white output area boundaries
tm_shape(masked_kde, bbox = bounding_box) + tm_raster("ud", style = "quantile", n
= 100, legend.show = FALSE, palette = "YlGnBu") +
tm_shape(Output.Areas) + tm_borders(alpha=.3, col = "white") +
tm_layout(frame = FALSE)
```



We can also create catchment boundaries from the kernel density estimations.

```
# compute homeranges for 75%, 50%, 25% of points, objects are returned as spatial
polygon data frames
range75 <- getverticeshr(kde.output, percent = 75)
range50 <- getverticeshr(kde.output, percent = 50)
range25 <- getverticeshr(kde.output, percent = 25)
```

With the ranges calculated we can then map them together in tmap. Notice that this tmap combines multiple layers. Remember the layer at the bottom of the list is the last one to be printed, and therefore will appear at the front of the graphic.

Upon first glance, the code looks quite complicated. So to summarise, each line does the following…

```
.   Create a grey background using the Output.Areas polygon with white borders
.   Plot the locations of houses using House.Points
.   Plot the 75% range, set attributes for border and fill (i.e. colour, transpare
ncy, line width)
.   Plot the 50% range, set attributes for border and fill (i.e. colour, transpare
ncy, line width)
.   Plot the 25% range, set attributes for border and fill (i.e. colour, transpare
ncy, line width)
.   The outside frame is removed
```

```
# the code below creates a map of several layers using tmap
tm_shape(Output.Areas) + tm_fill(col = "#f0f0f0") + tm_borders(alpha=.8, col = "wh
ite") +
tm_shape(House.Points) + tm_dots(col = "blue") +
tm_shape(range75) + tm_borders(alpha=.7, col = "#fb6a4a", lwd = 2) + tm_fill(alpha
=.1, col = "#fb6a4a") +
tm_shape(range50) + tm_borders(alpha=.7, col = "#de2d26", lwd = 2) + tm_fill(alpha
=.1, col = "#de2d26") +
tm_shape(range25) + tm_borders(alpha=.7, col = "#a50f15", lwd = 2) + tm_fill(alpha
=.1, col = "#a50f15") +
tm_layout(frame = FALSE)
```

Now try adding an addition range for the densest 10% of our data.

Whilst mapping the densities of house sales is reasonably interesting, this technique can be applied to all sorts of point data. You could, for example, get two sets of data and create two separate ranges to compare their distributions - i.e. two species of animals.

If you wish to save your raster files to your computer, you can use the `WriteRaster()` function as demonstrated below.

```
writeRaster(masked_kde, filename = "kernel_density.grd")
```

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)