4. Joining Data

When doing spatial data analysis, it is quite common to need to merge different data sets together. There are two main ways of doing this, firstly using the `merge()` command which will match attribute data in data frames, and secondly, using the match technique, which works with attribute data in shape files.

Merging Attribute Data in Data Frames

The `merge()` function allows us to take two data sets and combine them into one, based on a common variable. To test this, import the following data by running this command:

```
# Set working directory
setwd("M:/R work")
# Read data from the web
data <- read.csv("http://data.alex-singleton.com/r-helpsheets/4/example.csv",
    header = TRUE, skip = 1)
```

And to check that it has imported correctly, which is always a good idea, run:

```
data
```

Which should output:

```
    Name Age     Place                School
1   John  20 Liverpool Hillside High School
2 Rachel  21   Norwich   Colman High School
3  Helen  34 Liverpool Hillside High School
```

You now need to create another data frame which we will use as an example. You could create another csv file and import this; however, we will illustrate another way of achieving this by joining a series of vector lists.

```
# Create a person vector
Person <- c("Paul", "Mike", "John", "Helen", "Mia", "Leo", "Rachel")
# Create a favourite functions vector
Function <- c("merge()", "read.csv()", "colnames()", "ncol()", "length()", "getwd()",
    "save.image()")
# We can now join these two vectors into a new data frame of favourite
# functions
fav_fun <- data.frame(Person, Function)
# View the fav_fun
fav_fun
```

Which should look like this:

```
  Person     Function
1   Paul      merge()
2   Mike   read.csv()
3   John   colnames()
4  Helen       ncol()
5    Mia     length()
6    Leo      getwd()
7 Rachel save.image()
```

We now have two data sets; `data`, which contains a list of people, locations and schools and `fav_fun`, which contains a list including those people as well as additional people who have attended R workshops.

The next step is to combine the two. What we are going to do is select the people in the `fav_fun` data frame who also appear in the `test` data frame, and copy their favourite R function into a new data frame, along with all the information from `test`.

We will refer to the two data frames as `x` and `y`. The x data frame is `data`; and the y is `fav_fun`. In `x`, the column containing the list of people is called "Name", and in `y`, it is called "Person". The parameters of the merge function first accept the two table names, and then the lookup columns as `by.x` or `by.y`. You should also include `all.x=TRUE` as a final parameter. This tells the function to keep all the records in `x`, but only those in `y` that match.

```
People_And_Functions <- merge(data, fav_fun, by.x = "Name", by.y = "Person",
    all.x = TRUE)
```

To see what this command has done, type `People_And_Functions` to show the content of the new data frame. This should look like:

```
    Name Age     Place               School     Function
1  Helen  34 Liverpool Hillside High School       ncol()
2   John  20 Liverpool Hillside High School   colnames()
3 Rachel  21   Norwich   Colman High School save.image()
```

If the by column names were named the same in both `x` and `y` (e.g. both called "Name"), we could specify this more simply with `by="column name"` rather than `by.x` and `by.y`; and finally, a critical issue when making any join is assuring that the "`by`" columns are in the same format.

Match data in a Shapefile

The `match()` function works in a very similar way to `merge()` but can be used to append attribute data to a shape file. 'merge()' will often cause errors when working with spatial data frames.

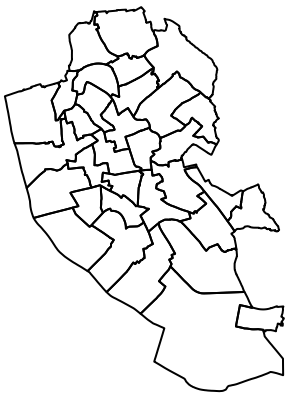
Load the required packages and example shapefile from helpsheet "3. Importing External Data".

```
library(rgdal)

# Set working directory
setwd("M:/R work")
# Download data.zip from the web
download.file("http://data.alex-singleton.com/r-helpsheets/4/data.zip", "data.zip")
# Unzip file
unzip("data.zip")

# Read in shape file
Wards <- readOGR(".", "england_caswa_2001")

# Plot Wards to check it has been imported correctly
plot(Wards)
```



We now have the content of the `Wards` shapefile in R. Have a look at the content of the data in the data slot:

```
head(Wards@data)
```

```
   gid ons_label        name  label
0  545    00BYGC  St. Mary's 04BYGC
1 2003    00BYFN      Dingle 04BYFN
2 2007    00BYFU Grassendale 04BYFU
3 2008    00BYFC    Allerton 04BYFC
4 2010    00BYFG  Broadgreen 04BYFG
5 2015    00BYFS    Gillmoss 04BYFS
```

We are now going to append the following data onto it, which are index scores for the rate of diabetes prevelance:

Ward	Rate
00BYGC	50
00BYFN	198
00BYFU	56
00BYFC	78
00BYFG	123
00BYFS	21

Run this code to create this data frame:

```
# Create an ons code vector
Ward <- c("00BYGC", "00BYFN", "00BYFU", "00BYFC", "00BYFG", "00BYFS")
# Create a rate vector
Rate <- c(50, 198, 56, 78, 123, 21)
# We can now join these two vectors into a new data frame of wards_diabetes
wards_diabetes <- data.frame(Ward, Rate)
# View the wards_diabetes
wards_diabetes
```

This should look like:

```
    Ward Rate
1 00BYGC   50
2 00BYFN  198
3 00BYFU   56
4 00BYFC   78
5 00BYFG  123
6 00BYFS   21
```

We can then use the `match()` function to append these diabetes rates on to `Wards@data`, by matching the `Ward` column from the `wards_diabetes` data frame to the `ons_label` column in the data slot of the wards SpatialPolygonsDataFrame.

```
Wards@data <- data.frame(Wards@data, wards_diabetes[match(Wards@data[, "ons_label"],
    wards_diabetes[, "Ward"]), ])
```

And to check, run:

```
head(Wards@data)
```

```
   gid ons_label        name  label   Ward Rate
0  545    00BYGC  St. Mary's 04BYGC 00BYGC   50
1 2003    00BYFN      Dingle 04BYFN 00BYFN  198
2 2007    00BYFU Grassendale 04BYFU 00BYFU   56
3 2008    00BYFC    Allerton 04BYFC 00BYFC   78
4 2010    00BYFG  Broadgreen 04BYFG 00BYFG  123
5 2015    00BYFS    Gillmoss 04BYFS 00BYFS   21
```

We have now appended the data, but also have the ward listed twice. To remove this, run:

```
Wards@data$Ward <- NULL
head(Wards@data)
```

Which changes `Wards@data` to:

```
   gid ons_label        name  label Rate
0  545    00BYGC  St. Mary's 04BYGC   50
1 2003    00BYFN      Dingle 04BYFN  198
2 2007    00BYFU Grassendale 04BYFU   56
3 2008    00BYFC    Allerton 04BYFC   78
4 2010    00BYFG  Broadgreen 04BYFG  123
5 2015    00BYFS    Gillmoss 04BYFS   21
```