Applied Research Methods

Workshop 3 - Exploring & Describing Data

Exploring data is a vital part of any analysis process. Not only does it provide a 'feel' for the nature of the data and the typical values present within it, but it can reveal trends and relationships that might be examined in further analysis. Both descriptive statistics and visualisation play a role in this - the first gives concrete values and allows for objective comparisons to be made, while the latter can often be more intuitive and convey a large amount of information quickly.

In this workshop, we will illustrate these ideas by exploring several aspects of some of the dataset that we used in Workshop 2. By the end of the workshop, you will have seen examples of various descriptive approaches which are generally applicable when performing quantitative analysis.

Describing categorical data

As a first step towards understanding the nature of crime in London, we'll take a look at the proportions of different crime types that occur. This can offer important insight, telling us - for example - about the relative frequency of property-based crime (e.g. theft) and violent crime (e.g. assault). Knowing which crimes are most common in a given setting not only tells us something about the kinds of crime to which the public is exposed, but also reflects the priorities of, and demands upon, the police.

We'll begin by examining the incident-level crime data that we downloaded in the previous Workshop. If you followed the instructions in that Workshop, you should have a file named 'NovIncidents.csv'. In R, make sure you have set your working directory correctly and then load these data by typing:

```
> data<-read.csv("NovIncidents.csv")
```

Check the data have loaded and look correct using the “head” command.

```
> head(data)
```

If you use the “summary” command, R will list the variables, indicate how many rows of data each variable has (this should be 91,204), what type of variable it is and, for numerical variables, provide some descriptive statistics.

```
> summary(data)
```

Let’s say we want to get a sense of what types of crime were recorded and with what frequency. The variable ‘crimetype' specifies the recorded crime type and we can use the ‘table’ command to summarise the data it contains. To do this, type:

```
> table(data$crimetype)
```

Recall that the "$" symbol is specified between the dataframe (in our case this is called "data") and the variable that is part of that dataframe that we are interested in. This should produce output like that below:

```
      Anti-social behaviour               Bicycle theft                    Burglary    Criminal damage and arson
                      18429                        1247                        6923                        4237
                      Drugs                Other crime                 Other theft        Possession of weapons
                       3614                         840                        9933                         478
              Public order                     Robbery                 Shoplifting        Theft from the person
                       4057                        3712                        3818                        4675
             Vehicle crime Violence and sexual offences
                      10358                       18883
>
```

The output tells us two things: 1) what types of crime were recorded; and, 2) with what frequency. For example, we can see that there were many incidents of Anti-social behaviour, but few "Possession of weapons" offences.

Visualising categorical data

Numerical frequencies are informative, but this information can also be expressed graphically. As we discussed in the lecture, graphical presentations can be more intuitive than numerical ones, particularly for making comparisons. In this case, since the variable we are examining is categorical, a natural way to express it is using a bar chart. To produce a barchart, we can either copy the values the "Table" command produced into new variables (one for the crime types and one for their frequencies), or we can (more easily) create a new dataframe using the "Table" command. To do this, type:

```
> crimes<-as.data.frame(table(data$crimetype))
```

To check that the dataframe appears to have been generated correctly, type its name:

```
> crimes
```

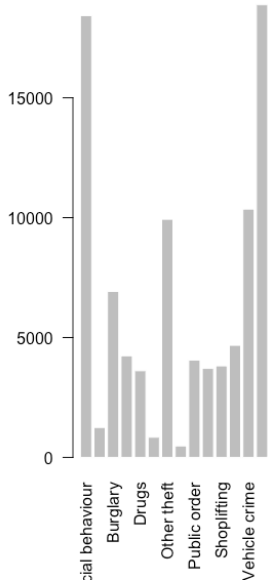
The output should look like the image below, and you will see that the dataframe has the two variables "Var1" and "Freq". The former contains the crime types, while the latter enumerates their frequency.

```
> crimes
                          Var1  Freq
1         Anti-social behaviour 18429
2                 Bicycle theft  1247
3                      Burglary  6923
4     Criminal damage and arson  4237
5                         Drugs  3614
6                   Other crime   840
7                   Other theft  9933
8          Possession of weapons   478
9                  Public order  4057
10                      Robbery  3712
11                  Shoplifting  3818
12         Theft from the person  4675
13                Vehicle crime 10358
14 Violence and sexual offences 18883
```

To produce a bar chart, we use the “barplot” command. In parentheses, we first specify the variable we want to summarise in the plot (“crimes$Freq”) and then note the variable that provides the labels for the bar chart (“crimes$Var1”). The other parameters shown determine whether the bars have borders (“border=F”) and how (using the “las” parameter) we want to orient the labels. If we set “las” to 2 the labels are shown perpendicular to the x-axis.

```
>barplot(crimes$Freq, names.arg=crimes$Var1, border=F ,las=2)
```

The plot should look like below.

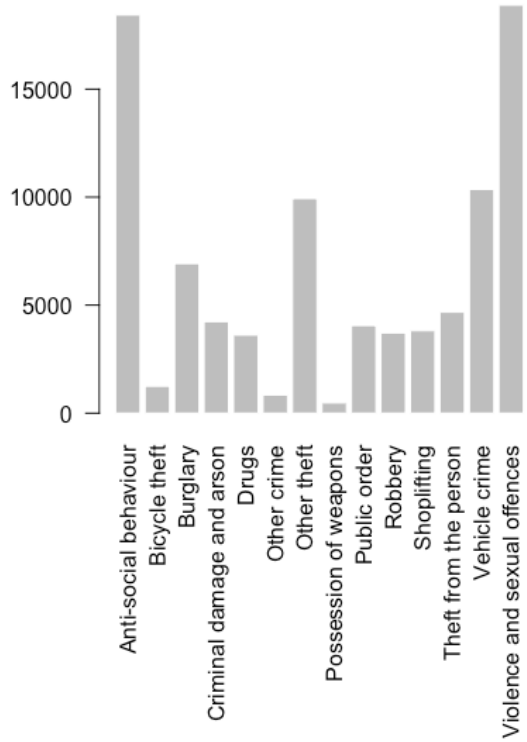


Try setting “las” to 1 and see how the figure looks. If you want to add an outline to the bars, set “border=T”. Looking at the figure above, one problem is that we cannot see the whole label for some crime types. To address this, we need to tell R to change the margins around the plot. To do this, we use the “mar” command. Type the following to change the margins and redraw the plot:

```
par(mar=c(14,4,4,4))
```

```
>barplot(crimes$Freq, names.arg=crimes$Var1, border=F ,las=2)
```

This makes the margin at the bottom of the plot larger (size=14). The values in brackets after the “mar” command specify how large each of the four margins should be (bottom, left, top, right). The plot should now look better:

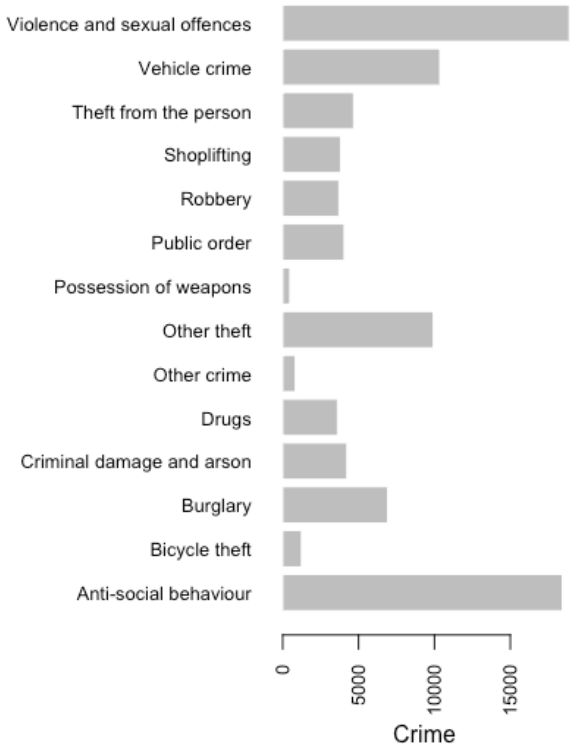


In this orientation the plot is hard to read, so let's rotate it using the "horiz" command. If we do this, the margins will be incorrect so we need to change the margins again. Try the following:

```
>par(mar=c(6,10,4,4))
```

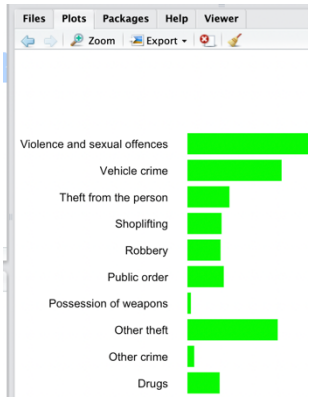
```
>barplot(crimes$Freq, names.arg=crimes$Var1, border=F ,las=2,
horiz=T, cex.names = 0.8, xlab="Crime", cex.axis=0.8)
```

This should produce the graph below. This is easier to read (to me at least). You should see some additional parameters were specified this time. These were "horiz" which is set to "T" which specifies that we want the barplot to be dispayed in this orientation; "cex.names = 0.8" which specifies the font size used for the axis labels (smaller values=smaller font); "xlab" which specifies what the x-axis label will be; and, "cex.axis=0.8" which specifies how large the font will be for the x-axis values (again, larger values=larger font).



Want to change the colour of the bars? Use the "col" command, for example, try:

```
> barplot(crimes$Freq, names.arg=crimes$Var1, border=F ,las=2,
horiz=T, cex.names = 0.8, xlab="Crime", cex.axis=0.8,
col="green")
```

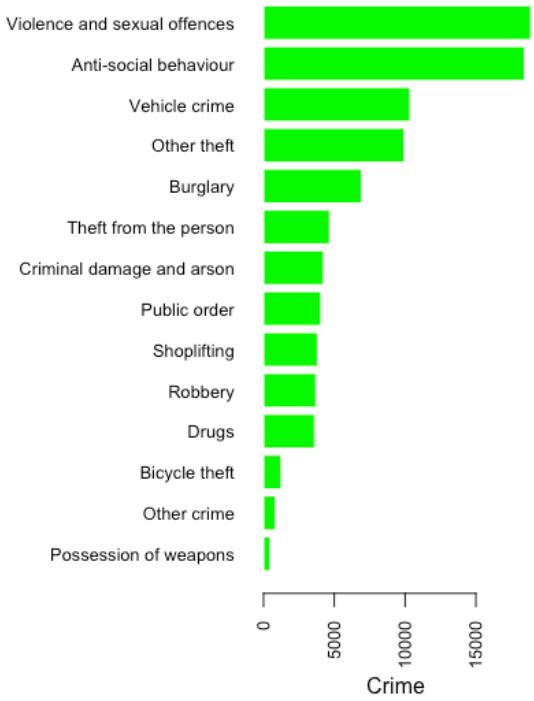


You can copy and paste graphics from R (or R Studio). If you are using R, you can just literally copy and paste. If you are using R Studio, from the plot window (see right), click "Export" and then either save the file or copy it to the clipboard (and then paste it). You can change the dimensions of the plot window at this stage.

Finally, in the plot above the graph is useful but it could work harder for the benefit of the reader. In particular, if we rank order the offence types, the reader can quickly understand which crime is most frequent and so on. To do this, when we create the dataframe that contains the table, we sort this, as follows:

```
> crimes<-as.data.frame(sort(table(data$crimetype)))
```

Reproducing the graph now generates a graph that is a little more helpful.



Describing numerical data

Having gained an impression of the nature of the crime in London, we'll move on to take a look at the extent of the problem, and in particular its spatial variation. To do this, we'll use the aggregated dataset that we constructed in Workshop 2, which contains crime counts for all LSOAs in London for November 2019.

Read the file, store it in a dataframe and use the 'head' command to check the data looks right. For example:

```
>LonNov<-read.csv("MergedData.csv")
```

```
>head(LonNov)
```

You should have a file with four variables and 4,825 observations. Recall that the variable "lsoacode" is the count of crimes in the corresponding LSOA (the name of each LSOA is stored in the variable "unique.values"). You might want to relabel these. You can do this in excel and reload the data or do this in R using the 'names' command, as follows:

```
>names(LonNov)[names(LonNov)=="lsoacode"] <- "Count"
```

```
>names(LonNov)[names(LonNov)=="unique.values"] <- "LSOA"
```

```
>head(LonNov)
```

In the above, the names shaded in light blue are the original variable names, those in yellow are the new variable names (these could be anything sensible that you choose). The text shaded green is the name of the dataframe.

The crime count represents the number of crimes that happened in each LSOA. While this count does give an indication of the level of crime, we know it can be misleading: some of the LSOAs may be larger than others, and if so we may expect their crime counts to be different even if there is no difference in the underlying level of offending. To address this, we need to convert the counts to **rates** - that is, standardise according to the size of the LSOAs. We'll do this using the population data that we added from the census last time.

To calculate the rates, we'll create a new variable in the dataframe. Type:

```
>LonNov$rate<- LonNov$Count/LonNov$HHLDS*1000
```

As discussed last week, the "<-" symbol assigns the value to the right of it to the variable to the left. Here we've added a new variable ("rate") to the existing dataframe ("LonNov"). We could have called it anything though. Use the 'head' command to check the new variable looks right. The rate is simply the crime volume divided by the count of households. We multiply this by 1000 so that the rate is expressed as a rate per 1000 homes. If we omit the multiplication by 1000, the numbers will be small and hard to understand. Use the "summary" command to get descriptive statistics:

```
> summary(LonNov$rate)
```

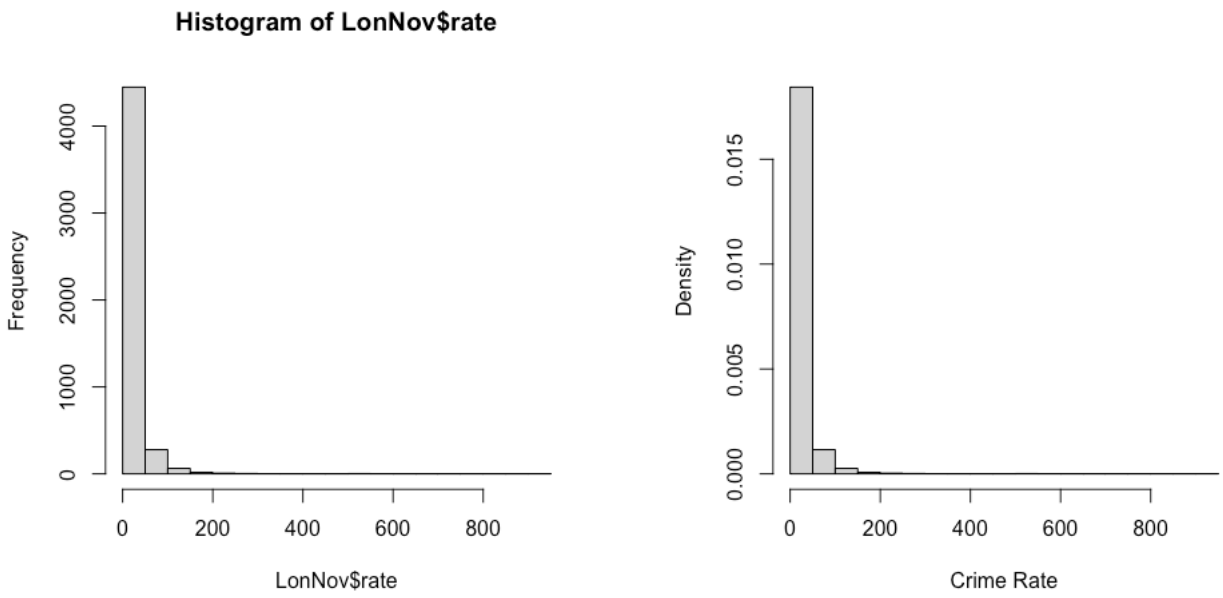
Visualising the distribution

To complement the numerical descriptive statistics, we can also look at the distribution of crime rates visually. Since 'rate' is a numerical variable, a histogram is a natural choice to do this. To plot the data, use the "hist" command:

```
> hist(LonNov$rate)
```

This will not produce the nicest looking graph (see left plot below) and by default will plot the actual frequencies rather than the relative frequencies. Try the following,

```
> hist(LonNov$rate, freq=FALSE, xlab="Crime Rate", main="")
```



For this version (right panel), the "freq" parameter specifies that we do not want to plot the raw frequencies. The "xlab" parameter allows us to name the x label anything we want, while the "main" parameter allows us to change the name of the plot. If we just insert "" we get white space. If you add text between the speech marks, this will be the plot title. In the above plot I generated the two graphs side by side. R will allow you to produce more than one graph in the same plot (which can make the figure easier to work with after you export it). To do this, you have to use the "mfrow" command. To reproduce the plot above, we tell R that we want one row of plots with two columns:

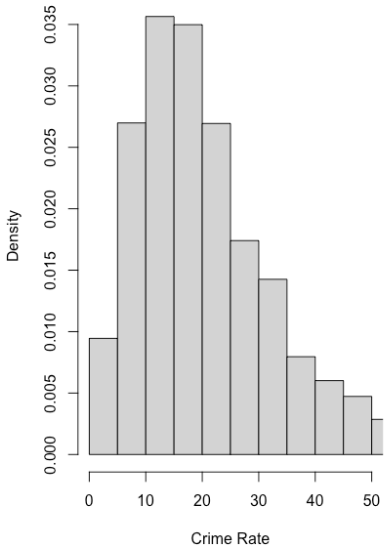
```
> par(mfrow=c(1,2))
```

The histogram that is generated doesn't reveal very much - there is a single tall spike on the left, followed by a couple of smaller ones, then only extremely small bars. The reason for this is that, even though almost all LSOAs have a crime rate less than 30, there are a small number with exceptionally high crime rates (i.e. outliers) - some as high as 947. Since R shows the full range of the distribution, this means it is difficult to see any detail at the lower end, where most of the values are.

We can address this in a number of ways. For example, we could select only the values within a certain range and then plot those. Here, we will address the issue by restricting the values that are plotted and by specifying the "bin" widths used in the plot. Try:

```
> hist(LonNov$rate, freq=FALSE, xlab="Crime Rate", main="",
xlim=c(0,50),seq(0,950,5), border=FALSE)
```

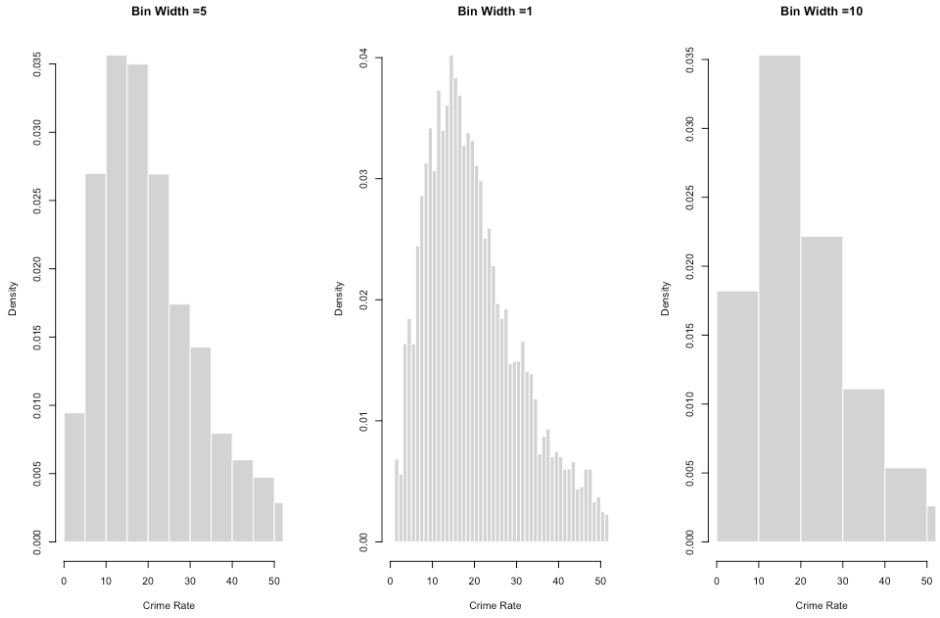
The "xlim" parameter specifies the range of values to be shown on the x-axis of the plot (0 to 50 in this case). The "seq" parameter specifies the range for the bins (0 to 950) and the interval to be used (increments of 5). The "border" parameter tells R not to draw an outline around the bars (contrast what you get to what is shown below).



The image is now much clearer. You can see that the distribution follows a curved shape, with a peak around the low 10s. You can see that most crime rates are below 30, and that the proportion gradually decreases for higher crime rates. This is consistent with the mean, median and quartiles that we calculated earlier.

> ***IMPORTANT**: While we have filtered the data to get a better visual impression, this does not mean that the outliers do not exist. The adjusted plot is useful in an exploratory sense, but we should still be mindful that some LSOAs have crime rates well outside this range - a distribution like this is referred to as 'long tailed' because its 'tail' stretches over a wide range. In particular, it would be misleading (or dishonest) to present the truncated plot as the distribution of crime rates without acknowledging the outliers.*

As an experiment, you may want to try modifying the 'bin widths' used for the plot. You might want to plot several versions at once, perhaps labelling the plots accordingly (see below).

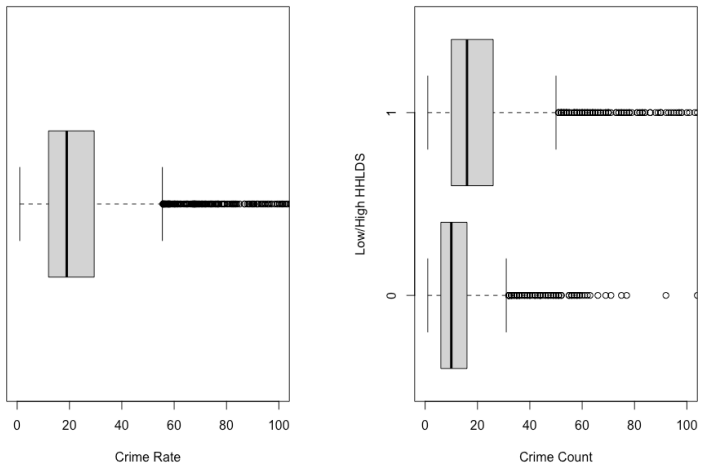


Comparing groups

An alternative to plotting the data is to use a boxplot. The left panel below was created using the following command. Note that the text in green specifies that we only want to plot the range of values between zero and 100. If we do not specify the "ylim" parameter, R will plot the full range of values. Note that the actual box plot is not affected by this – the command just essentially "crops" what we see.

```
>boxplot(LonNov$rate, horizontal = T, xlab="Crime Rate",
ylim=c(0,100))
```

This is useful, but the histogram is probably more helpful. One of the situations in which boxplots are perhaps more valuable is when we want to compare the distributions of two or more groups. The plot on the right below does this. Here, we show the boxplots for crime counts for those LSOAs that have a household count that is above the median (labelled 1) and below it (labelled zero).



To do this, I created a new variable which I called "Dummy". I initially assigned zero to all values of this variable.

```
>LonNov$Dummy<-0
```

I then assigned a value of 1 to all rows of the dataframe for which the household count was above the median value of 672.

```
>LonNov$Dummy[LonNov$HHLDS>672]<-1
```

I then used the "head" command to check that this had worked as expected.

```
>head(LonNov)
```

As you might expect, the distribution of crime counts for those LSOAs with more homes is shifted to the right of the distribution for those with less homes. Here is the command used to generate the second boxplot:

```
>attach(LonNov)
```

```
>boxplot(Count~Dummy, horizontal = T, xlab="Crime Count",
ylim=c(0,100), ylab="Low/High HHLDS")
```

To explain, the text in yellow indicates which variable is to be summarised (Count) and the "~" specifies that the data should be grouped by a specific variable (in this case "Dummy"). Everything else is as before.

Bivariate relationships

Having gained some insight into the distribution of single variable, we may begin to consider whether variables have (bivariate) relationships with each other. There may be good reasons to anticipate such relationships in some cases. In the simplest case, we might expect a relationship between crime and the number of households in an LSOA. Even in the absence of such hypotheses, exploring and visualising the associations in data can be a good way to uncover potential relationships.

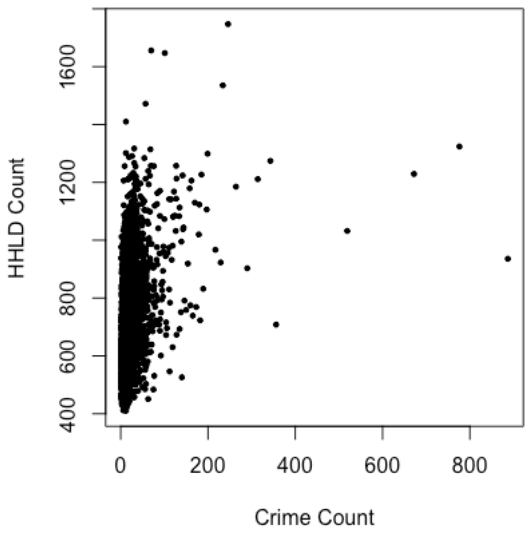
To demonstrate this, we'll examine the relationship between the crime count and the number of households in London's LSOAs. To do this, we will use the "plot" command to create a scatterplot. But first, if you only want to generate one plot in the plot window, you should reset this as follows:

```
> par(mfrow=c(1,1))
```

To generate the plot:

```
> plot(LonNov$Count, LonNov$HHLDS, pch=19, cex=0.5,
xlab="Crime Count", ylab="HHLD Count")
```

In the above command, the first variable (highlighted yellow) is the x variable, and the second (highlighted green) is the y variable to be plotted. The "pch" parameter specifies what type of shape we want to use to plot points. Try different values (e.g. 15). The "cex" value specifies the size of the points.



In closing

The aim of this workshop was to explore further ways of manipulating data in R , other ways of creating new variables and some approaches to plotting data. In introducing the commands used, we have discussed several parameters that are used by other commands and so the exercises completed above should be useful for understanding some commands that were not covered here. We have also used some general commands to control the plot window that should be of wider value. All of the commands used here are included in the R base package. There are other commands available in libraries that you can download to extend the capability of R, such as “ggplot” and “tidyverse”. To keep things simple, where possible, we will avoid their use, but if you are interested in using them, see the free resource https://r4ds.had.co.nz/index.html.