Practical 2: Data exploration in R

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

This practical will introduce you to the data exploration (http://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm) techniques which are useful for deriving an understanding of large numerical variables in R. We will introduce you to some useful R commands which allow you to observe the data efficiently and also create descriptive statistics. We will then visualise the distribution(s) of our data through the creation of univariate plots. 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:

- View and explore the data
- Create descriptive statistics
- Observe and compare the data using univariate plots
- Install and load an R package to use bespoke functions

Before we start, we need to do two things. First, we need to set the working directory.

```
# Set the working directory (remember to change this to your file path).
setwd("C:/Users/Guy/Documents/Teaching/CDRC/Practicals") # Note the single / (\\ will also work).
```

Second, load the data we saved in the previous practical.

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

Exploring the data

There are several ways to view data, some of been exemplified below. Remember R is case sensitive.To view the *Census.Data* object type:

```
# prints the data within the console
print(Census.Data)
```

We can also select which columns and rows we wish to print by entering the numerical ranges of the array within square brackets after the variable name (i.e. `Census.Data[n,n]`) where the comma separates the rows and columns. In this example, we are selecting rows 1 to 20 and columns 1 to 5. As we are selecting all of the columns in the data we could just leave the space after the comma in the square brackets blank.

```
# prints the selected data within the console
print(Census.Data[1:20,1:5])
```

```
##           OA White_British Low_Occupancy Unemployed Qualification
## 1  E00004120      42.35669     6.2937063  1.8939394      73.62637
## 2  E00004121      47.20000     5.9322034  2.6881720      69.90291
## 3  E00004122      40.67797     2.9126214  1.2121212      67.58242
## 4  E00004123      49.66216     0.9259259  2.8037383      60.77586
## 5  E00004124      51.13636     2.0000000  3.8167939      65.98639
## 6  E00004125      41.41791     3.9325843  3.8461538      74.20635
## 7  E00004126      48.54015     5.5555556  4.5454545      62.44726
## 8  E00004127      48.67925     8.8709677  0.9389671      60.35242
## 9  E00004128      45.39249     2.4844720  2.1645022      70.07874
## 10 E00004129      49.05660     3.5211268  4.3103448      66.66667
## 11 E00004130      38.80597     6.2500000  0.9174312      66.66667
## 12 E00004131      39.64286     7.5630252  1.8691589      64.47368
## 13 E00004132      55.88235     4.3478261  3.7974684      73.49398
## 14 E00004133      41.96078     7.6271186  1.9900498      65.38462
## 15 E00004134      53.19149     6.0000000  2.7027027      72.89157
## 16 E00004135      46.85315     4.7619048  3.7313433      74.82014
## 17 E00004136      59.64912     0.9090909  2.7322404      73.68421
## 18 E00004137      48.16176     5.4421769  2.7522936      69.06780
## 19 E00004138      42.22222     2.8169014  4.9723757      58.16327
## 20 E00004139      17.71772    64.2857143 15.9420290      22.96651
```

You can also open the data in R using the `View()` function. This will create a clearly formatted table in a new window which displays the top 1000 cases.

```
# to view the top 1000 cases of a data frame
View(Census.Data)
```

If the data is very large and opening it could be computationally intensive - you could just opt to open the top or bottom *n* cases.

The `head()` and `tail()` commands open the top and bottom *n* cases respectively.

```
head(Census.Data)
```

```
##          OA White_British Low_Occupancy Unemployed Qualification
## 1 E00004120      42.35669     6.2937063   1.893939      73.62637
## 2 E00004121      47.20000     5.9322034   2.688172      69.90291
## 3 E00004122      40.67797     2.9126214   1.212121      67.58242
## 4 E00004123      49.66216     0.9259259   2.803738      60.77586
## 5 E00004124      51.13636     2.0000000   3.816794      65.98639
## 6 E00004125      41.41791     3.9325843   3.846154      74.20635
```

```
tail(Census.Data)
```

```
##              OA White_British Low_Occupancy Unemployed Qualification
## 744 E00174675     37.354086      9.401709   2.714932      52.81385
## 745 E00174676      7.881773      9.868421   0.500000      37.12871
## 746 E00174677     22.520107      8.125000   4.528302      50.67568
## 747 E00174678     23.949580      6.194690   1.421801      53.21101
## 748 E00174679     24.271845      4.081633   1.663894      45.34884
## 749 E00174680     36.514523     25.274725   8.108108      24.74227
```

It is also easy to observe the number of rows and columns and the names (or headers) of each of the columns.

```
#Get the number of columns
ncol(Census.Data)
```

```
## [1] 5
```

```
#Get the number of rows
nrow(Census.Data)
```

```
## [1] 749
```

```
#List the column headings
names(Census.Data)
```

```
## [1] "OA"            "White_British" "Low_Occupancy" "Unemployed"
## [5] "Qualification"
```

Whilst it is informative to open data, it is often difficult to generalise key trends just by looking at the numbers.

Descriptive statistics

Descriptive statistics are a useful means of deriving quick information about a collective dataset. A good introduction to descriptive statistics is available in the online version of Statistical Analysis Handbook (de Smith, 2015) (http://www.statsref.com/HTML/?descriptive_statistics.html) which includes detailed descriptions of measures of central tendecy (http://www.statsref.com/HTML/?averages.html) and measures of spread (http://www.statsref.com/HTML/?measures_of_spread.html)

Notice that below we use the $ symbol to select a single variable from the *Census.Data* object. If you type in `Census.Data$` (so the name of your data object followed by a $ sign), then press tab on your keyboard, RStudio will let you select a variable from a drop down window. Repeat this step for your qualifications variable.

```
mean(Census.Data$Unemployed)
```

```
## [1] 4.510309
```

```
median(Census.Data$Unemployed)
```

```
## [1] 4.186047
```

```
range(Census.Data$Unemployed)
```

```
## [1]  0.00000 18.62348
```

A useful function for descriptive statistics is `summary()` which will produce multiple descriptive statistics as a single output. It can also be run for multiple variables or an entire data object.

```
#mean, median, 25th and 75th quartiles, min, max
summary(Census.Data)
```

```
##          OA      White_British    Low_Occupancy      Unemployed
##  E00004120:  1   Min.   : 7.882   Min.   : 0.000   Min.   : 0.000
##  E00004121:  1   1st Qu.:35.915   1st Qu.: 6.015   1st Qu.: 2.500
##  E00004122:  1   Median :44.541   Median :10.000   Median : 4.186
##  E00004123:  1   Mean   :44.832   Mean   :11.597   Mean   : 4.510
##  E00004124:  1   3rd Qu.:54.472   3rd Qu.:16.107   3rd Qu.: 6.158
##  E00004125:  1   Max.   :78.035   Max.   :64.286   Max.   :18.623
##  (Other)  :743
##  Qualification
##  Min.   :11.64
##  1st Qu.:36.32
##  Median :55.10
##  Mean   :51.43
##  3rd Qu.:66.23
##  Max.   :88.07
##
```

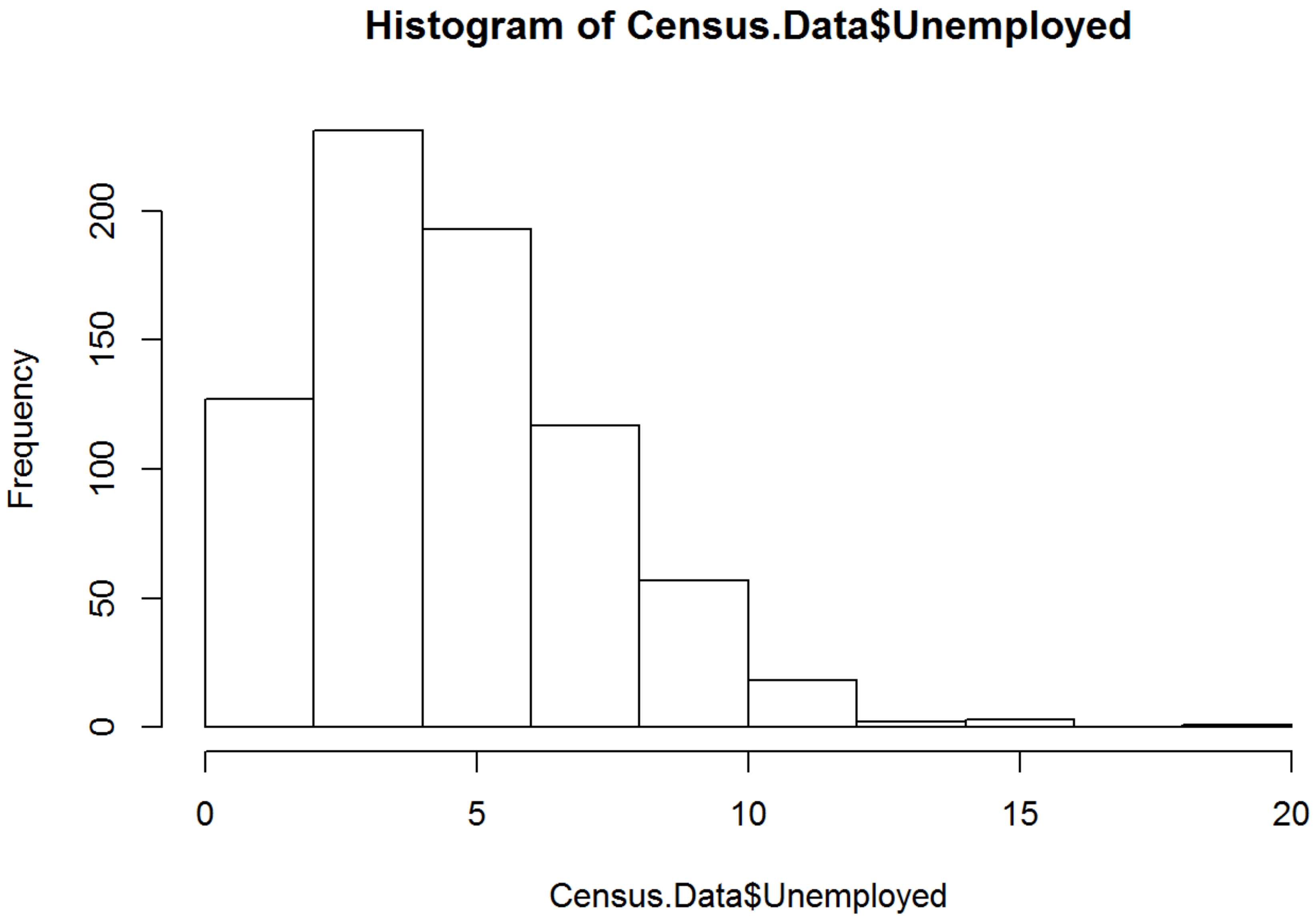
Univariate plots

Univariate plots are a useful means of conveying the distribution of a particular variable. Many of these can be produced very simply in R.

Histograms

Histograms (http://www.itl.nist.gov/div898/handbook/eda/section3/histogra.htm) are perhaps the most informative means of visualising a univariate distribution. In the example below we will create a histogram for the *unemployed* variable using the `hist()` function (remember if you put a $ symbol followed by the column header after the data object, the function in R will read that column only). Repeat this step for your qualifications variable.

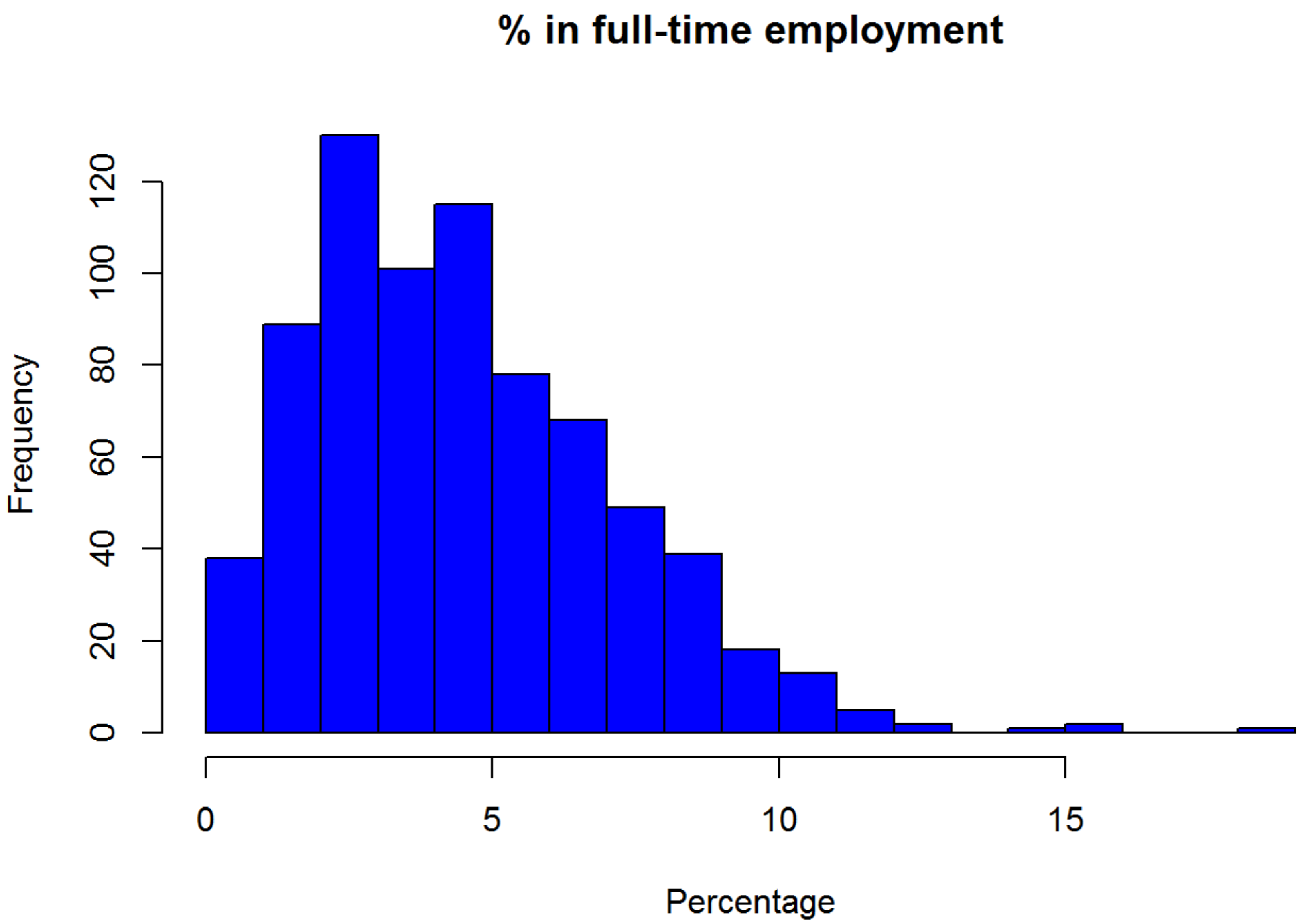
```
# Creates a histogram
hist(Census.Data$Unemployed)
```



The histogram should appear in the *Plots* window of RStudio.

We can tidy up the histogram by including the following parameters within the `hist()` function. In the example below, we specify the number of data breaks in the chart (`breaks`), colour the chart blue (`col`), create a title (`main`) and label the x-axis (`xlab`). For more information on all of the parameters of the function just type `?hist` into R and run it.

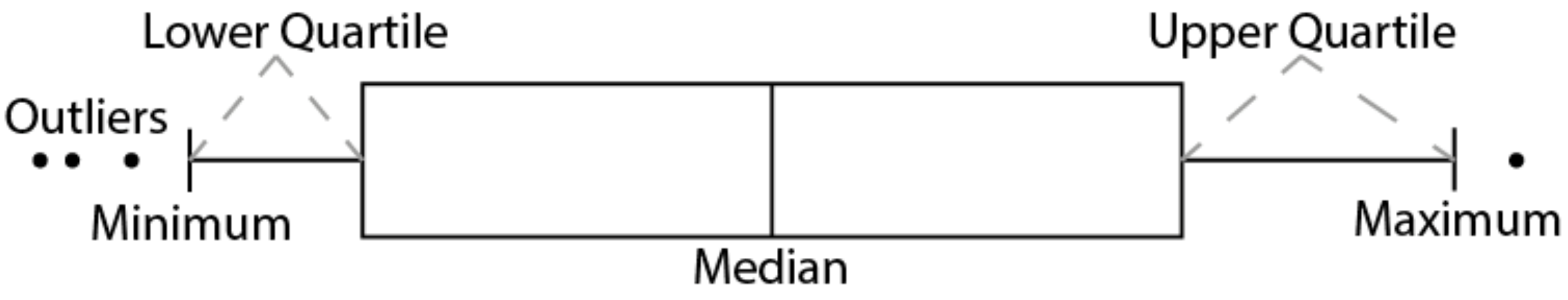
```
# Creates a histogram, enters more commands about the visualisation
hist(Census.Data$Unemployed, breaks=20, col= "blue", main="% in full-time employme
nt", xlab="Percentage")
```



Notice that in the example above, we have specified that there are 20 breaks (or columns in the graph). The higher the number of breaks, the more intricate and complex a histogram becomes. Try producing a histogram with 50 breaks to observe the difference.

Boxplots

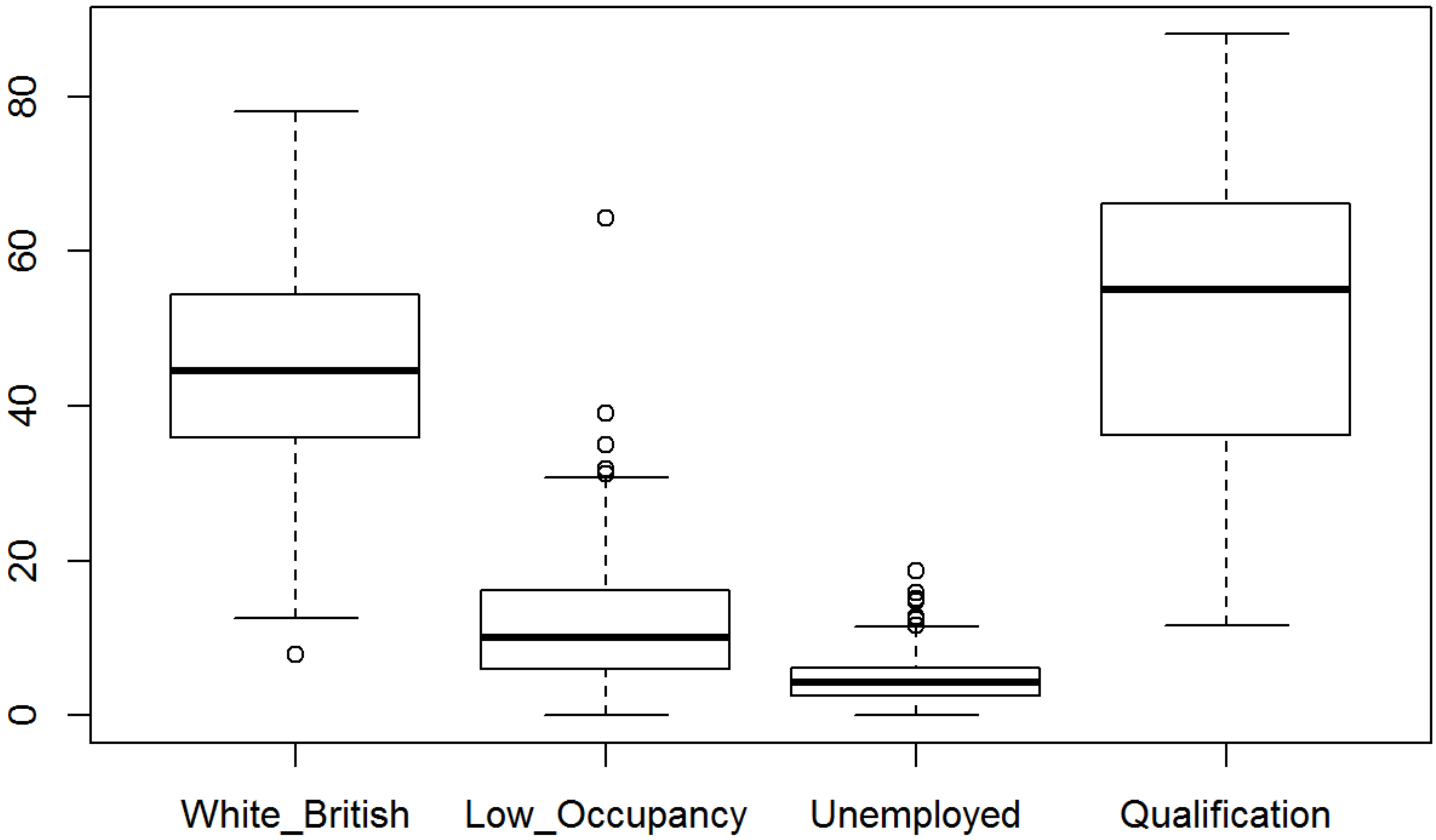
In addition to histograms, another type of plot that shows the core characteristics of the distribution of values within a dataset, and includes some of the `summary()` information we generated earlier, is a box and whisker plot (box plot (http://www.statsref.com/HTML/?measures_of_spread.html) for short).



Box plots can be created in R by sampling running the `boxplot()` function. For more information on the parameters of this function type `?boxplot` into R.

In the example below, we are creating multiple box plots in order to compare the four main variables. To select the variables we have used an array after *Census.Data* so that the function doesn't read the row names too. We could also call individual rows i.e. `boxplot(Census.Data$Unemployed)` or even pairs of variables i.e. `boxplot(Census.Data$Unemployed, Census.Data$Qualification)`.

```
# box and whisker plots
boxplot(Census.Data[,2:5])
```



Installing the vioplot package and creating a violin plot

One of the main benefits of R is that as an open source tool, there is much documentation on how to complete more advanced tasks online. In addition to R's core functions, there are also a large volume of bespoke packages which include their own niche functions. These packages can be downloaded and installed for free using R.

In this case, we want to create a type of univariate plot known as a violin plot. In its simplest form, a violin plot combines both box plots and histograms in one graphic. It is also possible to input multiple plots in a single image in order to draw comparisons. However, there is no function to create these plots in the standard R library so we need to install a new package.

Step 1: Install the package

To install go to **Tools > Install packages.** in RStudio and enter *vioplot*. Alternatively, run `install.packages()` as demonstrated below,

```
# When you hit enter R will ask you to select a mirror to download the package contents from. It doesn't really matter which one you choose, I tend to pick the UK based ones.

install.packages("vioplot")
```

The `install.packages` step only needs to be performed once. You don't need to install a package every time you want to use it. However, each time you open R and wish to use a package you need to use the `library()` command to tell R that it will be required.

Step 2: Open the package

To ensure that R connects to the package and the new functions are activated you need to activate the package. This can be done using the `library()` or `require()` packages. Simply enter the name of the downloaded package within the brackets of either function.

```
# loads a package
library(vioplot)
```

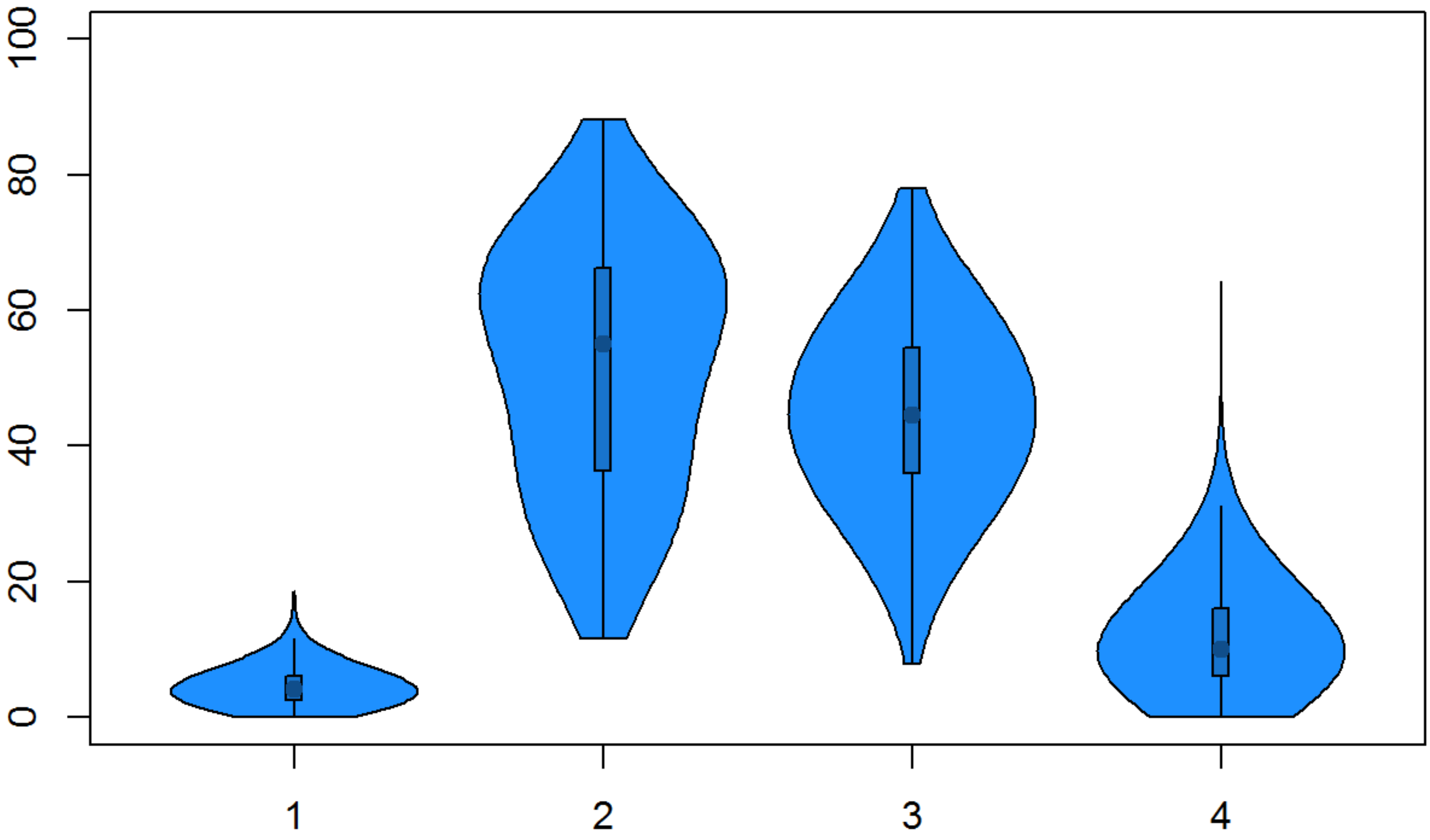
```
## Loading required package: sm
```

```
## Package 'sm', version 2.2-5.4: type help(sm) for summary information
```

Step 3: Using newly installed functions

Now we are ready to use the newly available `vioplot()` function. Remember you can run `?vioplot` to explore the parameters of the function. In its very simplest form, you can just write `vioplot(Census.Data$Unemployed)` to create a simple plot for the Unemployed variable. However, the example below includes all four variables and a couple of extra parameters. The `ylim` command allows you to set the upper and lower limits of the y-axis (in this case 0 and 100 as all data is percentages). Three unique colours were also assigned to different parts of the plot.

```
# creates a violin plot for 4 variables, uses 3 shades of blue
vioplot(Census.Data$Unemployed, Census.Data$Qualification, Census.Data$White_British, Census.Data$Low_Occupancy, ylim=c(0,100), col = "dodgerblue", rectCol="dodgerblue3", colMed="dodgerblue4")
```

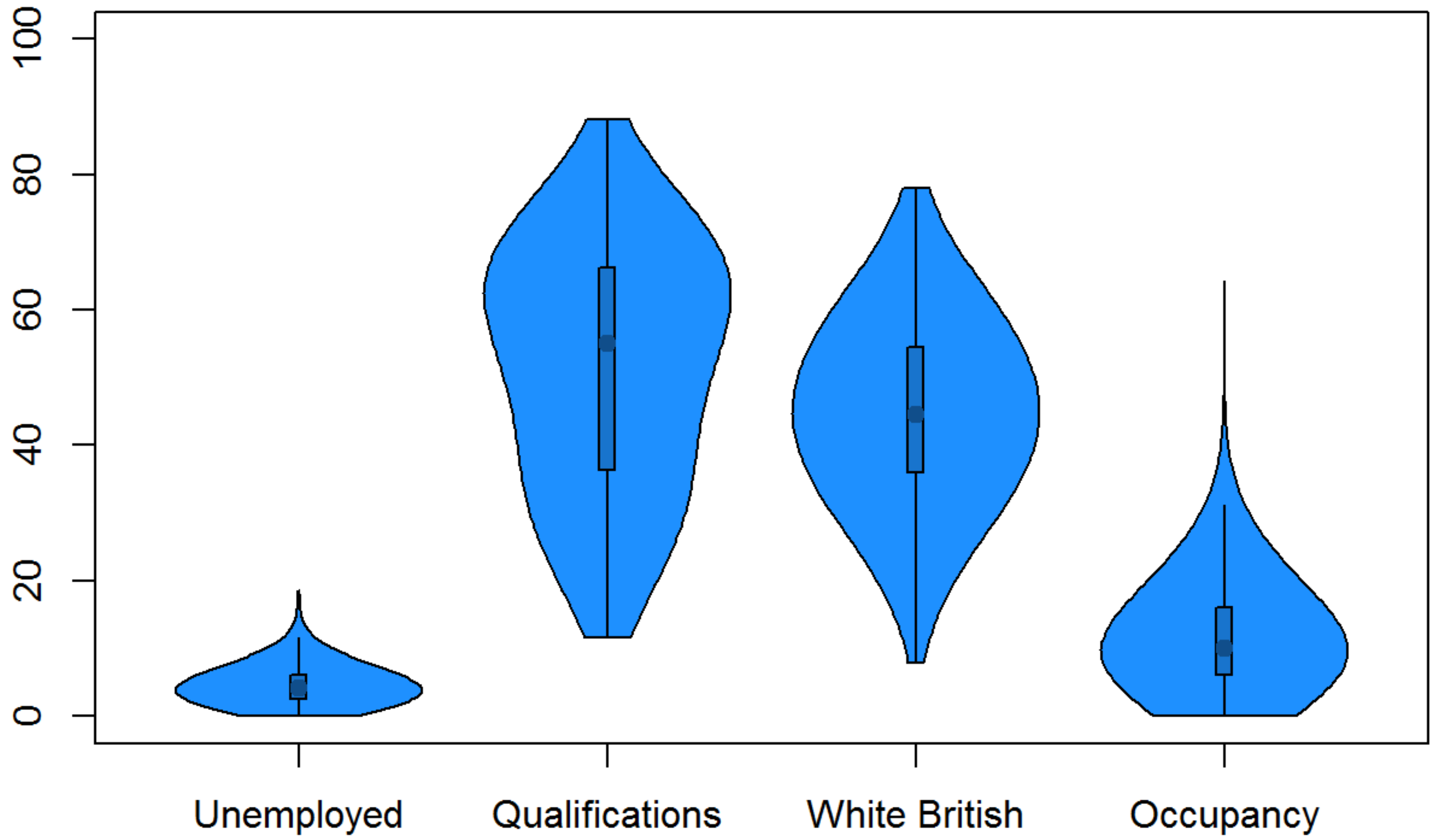


Recreate the violin plots using different colours. Colours can be specified in various different forms such as predefined names (as demonstrated above) or using RGB or HEX colour codes. This PDF outlines the names of many colours in R and may be useful for you: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf (http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf).

For more information on graphical parameters please see: http://www.statmethods.net/advgraphs/parameters.html (http://www.statmethods.net/advgraphs/parameters.html)

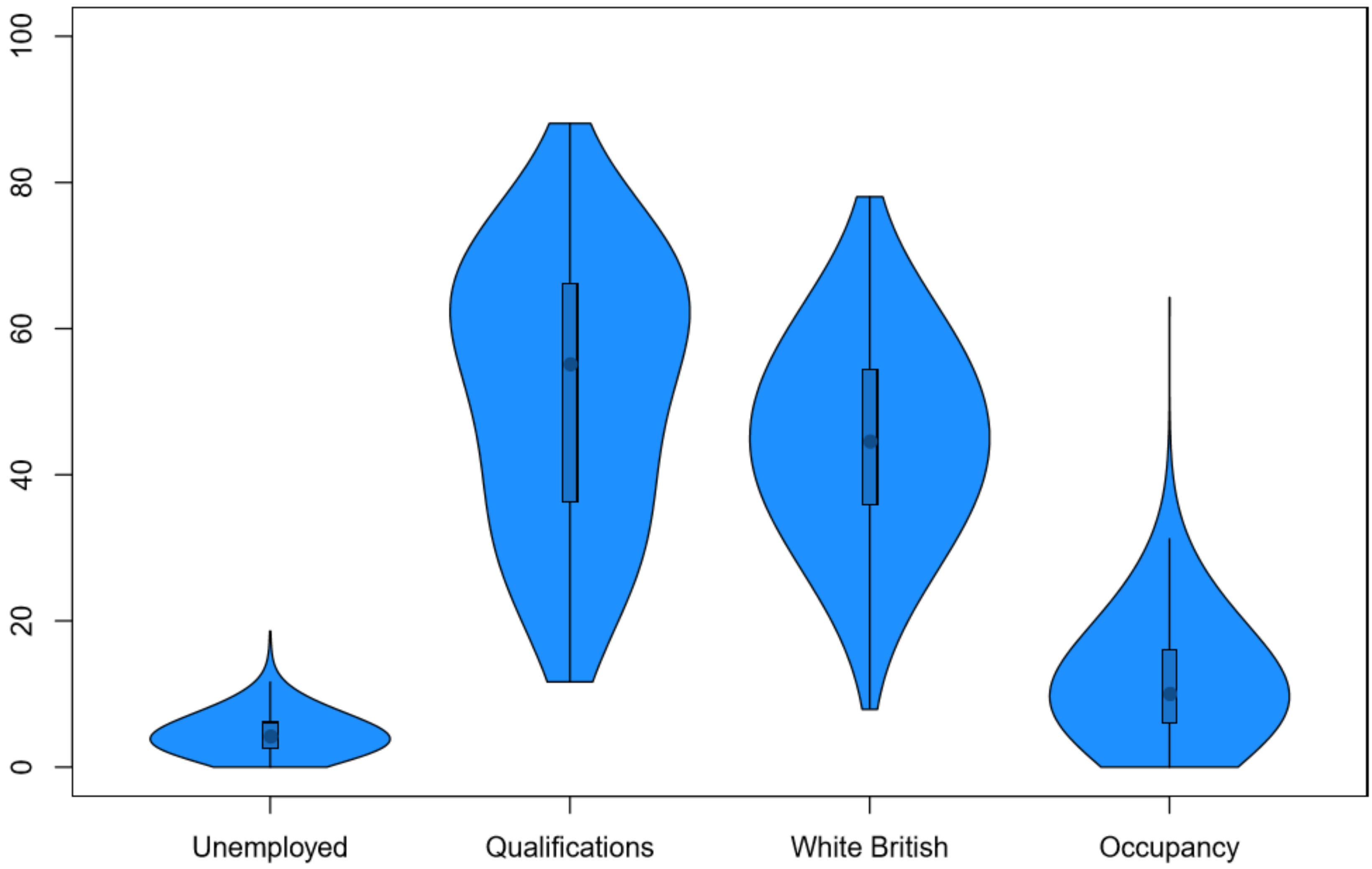
Finally, create labels for each of the data in the graphic. We can do this using the names command within the `vioplot()` function as demonstrated below.

```
# add names to the plot
vioplot(Census.Data$Unemployed, Census.Data$Qualification, Census.Data$White_British, Census.Data$Low_Occupancy, ylim=c(0,100), col = "dodgerblue", rectCol="dodgerblue3", colMed="dodgerblue4", names=c("Unemployed", "Qualifications", "White British", "Occupancy"))
```



Exporting images

You can export the images from R if you want to observe them in a much higher quality. You can do this by clicking on **Export** within the **Plots** window in RStudio. PDF versions are exported in a very high quality. Below is an example of an exported version of the image above.



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)