

Week 2 Practical

Getting Started with Data Exploration


Overview

In this Practical we will introduce some of the commonly used data exploration techniques in R. We will focus on several useful R commands which allow you to observe data and prepare descriptive statistics.

Credit: The practical was developed following CRDC Data (<https://data.cdrc.ac.uk/>) tutorials.

How to use this practical?

The practical provides a set of tasks you should follow to learn a set of skills you will need for you assignments. This is by no means intended to be a complete guide to all the functions and methods available out there for a specific task. It rather focuses on commonly used methods and a minimum you need to understand to be successful in your assignments.

Icon  indicates a challenge you should try to solve. Those challenges are highly recommended.

Prerequisites:

- Installed R and RStudio.
- Previous experience with R is desirable, not mandatory.

Objectives:

- Load and explore data
- Create descriptive statistics
- Observe data using univariate plots

Sharing results

While tasks should be self-explanatory, some of the challenges might be more complex. Feel free to share your results or question in the course discussion forum. Results to all the challenges will be available UPON REQUEST (discussion forum or email).

Dataset

We will be using data from CDRC 2011 Census Geodata pack - Camden (E09000007) - <https://data.cdrc.ac.uk/dataset/introduction-spatial-data-analysis-and-visualisation-r/resource/practical-data>. Table 1 provides an overview of the available columns.

Table 1. Camden census data description

Column Name	Description	Data type
OA	Output Area - Output areas (OA) were created for Census data, specifically for the output of census estimates. The OA is the lowest geographical level at which census estimates are provided.	Nominal
White_British	The percentage of White British population in the output area	Numeric, continuous
Low_Occupancy	The percentage of low-occupied households in the output area	Numeric, continuous
Unemployed	The unemployment rate for the output area	Numeric, continuous
Qualification	The percentage of residents who have Level 1 qualifications or above.	Numeric, continuous

Before getting started

Before going through the tasks, make sure you have a new R Notebook created in RStudio, as shown in Figure 1.

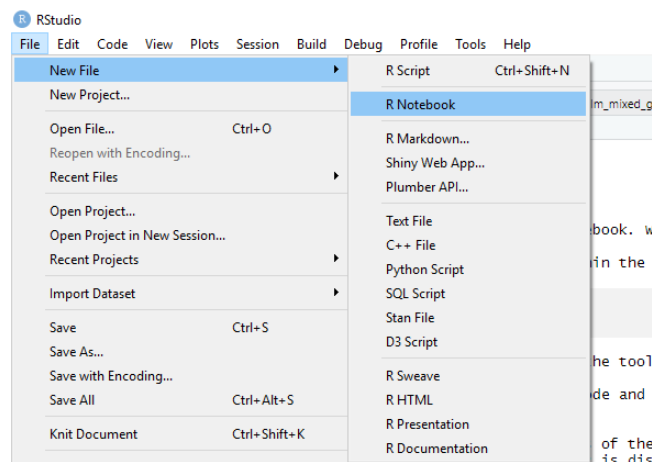


Figure 1. Opening R Notebook in RStudio

Each section of the code (marked by the grey box) should be executed within a single chunk in your Notebook.

Task 1. Load data

There are several ways you can load your data. In this example, you can load data from the URL: http://bit.ly/infs5100_camden_data

```
census.data <- read.csv(url("http://bit.ly/infs5100_camden_data"))
```

Alternatively, you can download `camden_census_data.csv` from learnonline and load data from your drive.

Task 2. Explore data

Once you load the data, it is a common practice to explore your dataset and make sure everything went well with the import.

You can display the loaded dataset (showing the top 1000 cases) using the `View()` function.

```
View(census.data)
```

Usually, it is more than enough to show the top or bottom n cases. The `head()` and `tail()` commands open the top and bottom n cases, respectively.

```
head(census.data)
```

```
tail(census.data)
```

By default, both methods return six (6) cases, either top or bottom. However, you can explicitly assign an exact number of cases to be returned.

```
head(census.data, n=10)
```

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)
```

```
#Get the number of rows  
nrow(census.data)
```

```
#List the column headings  
names(census.data)
```



Challenge 1. Can you rename columns in your dataset as follows?

- OA into `Output_Area`,
- `White_British`, `Low_Occupancy`, `Unemployed` and `Qualification` should remain the same.

The output of the `names()` command should be as below.

```
#List the column headings  
names(census.data)
```

```
[1] "Output_Area" "White_British" "Low_Occupancy" "Unemployed" "Qualification"
```

Task 3. Extract descriptive statistics

Descriptive statistics are useful in deriving quick information about a collective dataset.

Here we will calculate some basic statistics – i.e., mean, median, standard deviation, and range – for one of the variables, before providing a summary for the entire dataset.

```
#Calculate mean for Unemployed  
mean(census.data$Unemployed)
```

```
#Calculate median for Unemployed  
median(census.data$Unemployed)
```

```
#Calculate standard deviation for Unemployed  
sd(census.data$Unemployed)
```

```
#Calculate range for Unemployed  
range(census.data$Unemployed)
```

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)
```



Challenge 2. Calculate summary statistics – mean, median, and variance - using the `summaryBy()` function from the `doBy()` package.

Package and function documentation can be found here:

<https://www.rdocumentation.org/packages/doBy/versions/4.6.8/topics/by-summary>

Task 4. Univariate plots

Univariate plots are a useful means of conveying the distribution of a particular variable.

Histograms are perhaps a most informative means of visualizing a univariate distribution. In the example below we will create a histogram for the **Unemployed** variable using the `hist()` function.

Repeat this step for your White British, Low Occupancy, and Qualifications variables.

```
# Creates a histogram  
hist(census.data$Unemployed)
```

We can format the histogram by including some of the available parameters within the `hist()` function. In the example below, we specify the number of data breaks in the chart (`breaks`),

color 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
```

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.

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 – or **Boxplots**.



Figure 2. Boxplots represent one of the most commonly used visualization techniques

Box plots can be created in R by 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 to compare the four main variables. To select the variables, we have used an array after `census.data` so that the function does not read the `Output_Area` column as well.

```
# box and whisker plots
boxplot(census.data[,2:5]), xlab="Percentage")
```

Finally, we will introduce another very useful visualization tool - **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.

Before plotting, you might need to install the `vioplot` package.

```
# install the vioplot package
install.packages("vioplot")
```

Once the installation is complete, you will have to load the newly installed package:

```
# install the vioplot package
library(vioplot)
```

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 colors were also assigned to different parts of the plot.

```
# 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")
```

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"))
```



Challenge 3. Can you export those plots in PDF or PNG format?

Citation and Copyright

Contains National Statistics data Crown copyright and database right 2020.

Contains Ordnance Survey data Crown copyright and database right 2020.