Practical 1: An Introduction to R

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

This practical is the first in a series of tutorials which will introduce you to a range of useful techniques for handling, analysing and visualising spatial data in R. All of the data and software used are freely available as open data and open software.

> "Open data is data that can be freely used, re-used and redistributed by anyone - subject only, at most, to the requirement to attribute and sharealike." *The Open Data Handbook (http://opendatahandbook.org/)*

Prior to working in R, we will first take you through obtaining the data from the Consumer Data Research Centre (https://www.cdrc.ac.uk/)'s (CDRC) data portal which stores consumer-related data from a large number of sources within the UK. Consumer data are data generated by retailers and other service organisations as part of their routine business processes. They are commonly used within the private sector to monitor the needs, preferences and behaviours of customers. However, the data is also of high value to public institutions. Census data is also useful to consumer insight, all leading retailers rely on accurate spatial data on population in order to guide the planning of their store locations and how their stock is distributed and marketed across the country.

The CDRC (https://www.cdrc.ac.uk/) offers an open data service through which members of the public can register online and freely download data pertaining to consumers. The service can be accessed by visiting https://data.cdrc.ac.uk/ (https://data.cdrc.ac.uk/)

The following practicals will use the London Borough of Camden as their case study area.

In this tutorial we will:

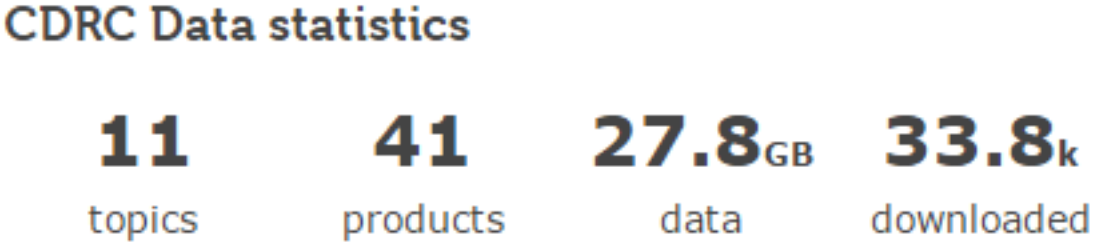
- Download a Census data pack from the CDRC Data website
- Load the data into R using RStudio
- View the raw data in R
- Subset data in R
- Merge data in R

Downloading data from the CDRC data website

Before we introduce you to R and Rstudio, we will first download some data from the CDRC Data Service. On an internet browser go to https://data.cdrc.ac.uk/ (https://data.cdrc.ac.uk/)

In the top right of the screen you will see options to log in or register for an account. If you have not yet registered please sign up to an account now.

We are going to be interested in small area Census data for the Borough of Camden. There are several routes to this dataset which include a search bar at the top of each page.



On the tab panel at the top of the page, click on **Topics**.

The data available on the CDRC data website can be broadly grouped into 11 key themes. Most of these pertain to the population and human activities. All of these themes are important to a wide range of industries, notably including retail. Census data can be found by clicking on the **Demographics** topic.

However, within this option there is still a very large number of datasets as the CDRC stores individual data packs for every district within the UK. In the search bar at the top of the page, enter ‘Camden’.

Demographics

Data pertaining to general consumer characteristics such as age and gender. Our collection also includes open source geodemographic classifications, such as the Output Area... read more

Followers 1

Datasets 5k

Follow

Tags

Demographics (5061)

2011 (2518)

Population (2217)

Estimates (2217)

2013 (2185)

Datasets | Activity Stream | About

Search datasets...

5,063 datasets found

Order by: Relevance

Small-Area Population Change 2011-14 Open

Small-Area Population Estimates from the ONS and NRS, for GB, at LSOA/DZ level, for 2011 and 2014.

CSV

CDRC 2011 Population Weighted Centroids - GB Open

The CDRC 2011 Population Weighted Centroids (LSOA/Data Zone) - GB data pack is integral of data from multiple sources which renders population weighted centroids for each LSOA...

CSV ZIP

Synthetic Population Safeguarded

This synthetic population contains individual level data from England and Wales, generated from the 2011 census data. Individual row level data is available for MSOA geography....

DOC

2011 COWZ-EW Geodata Pack - England and Wales Open

The 2011 Classification Of Workplace Zones (2011 COWZ-EW) is a UK geodemographic classification produced as a collaboration between the Office for National Statistics and the...

ZIP CSV PDF

You now want to scroll down to **CDRC 2011 Census Data Packs for Local Authority District: Camden (E09000007)**. It can also be found by clicking on the Census tab on the side of the screen.

The following page describes the content of the data and disclosure controls. The data is provided as a zipped folder of several different tables of Census data which encompass a wide variety of variables on the population. In addition, data is also provided at three different geographic scales of data units - Output Area, Lower Super Output Area and Middle Super Output Area. The geographic units have been described on the ONS website (http://www.ons.gov.uk/methodology/geography/ukgeographies/censusgeography).

To download the file you must click on **Camden.zip** under Data and Resources. If you have not already done so, here you will be asked to freely register as a new user.

It is recommended that you move the downloaded **Camden.zip** file to somewhere appropriate in your directory and then unzip the folder. To unzip on a windows computer simply right-click on the zipped file in windows explorer and click on **“Extract All.”**.

The folder includes lots of useful data. The **tables** subfolder is where the Census data is stored in its various forms. However, each table is given a code name which is not informative to us, so the **datasets_description** file is provided so we can lookup their names. In addition to this, a **variables_description** is provided so we can look up the variable name codes within each data table. GIS shapefiles (https://en.wikipedia.org/wiki/Shapefile) are also available from the **shapefiles** subfolder, these will be useful for mapping the data.

The Census data pack includes a large number of different datasets, all stored as Comma Separated Values (.csv) files. CSVs are a simple means of storing data so that it can be easily read and written on a computer. They are simply plain text documents where commas are used to separate the fields of data (you can observe this if you open a CSV in notepad).

For the forthcoming practicals we will be considering three variables, each from a different census dataset. These will be:

ColumnVariableCode	ColumnVariableDescription	DatasetId	DatasetTitle
KS201EW0020	White: English/Welsh/Scottish/Northern Irish/British	KS201EW	Ethnic group
KS403EW0012	Occupancy rating (bedrooms) of -1 or less	KS403EW	Rooms, bedrooms and central heating
KS501EW0019	Highest level of qualification: Level 4 qualifications and above[1]	KS501EW	Qualifications and students
KS601EW0019	Economically active: Unemployed	KS601EW	Economic activity

[1]Level 4 qualifications refer to a Certificate of Higher Education Higher National Certificate (awarded by a degree-awarding body).

Loading data and data formatting in R

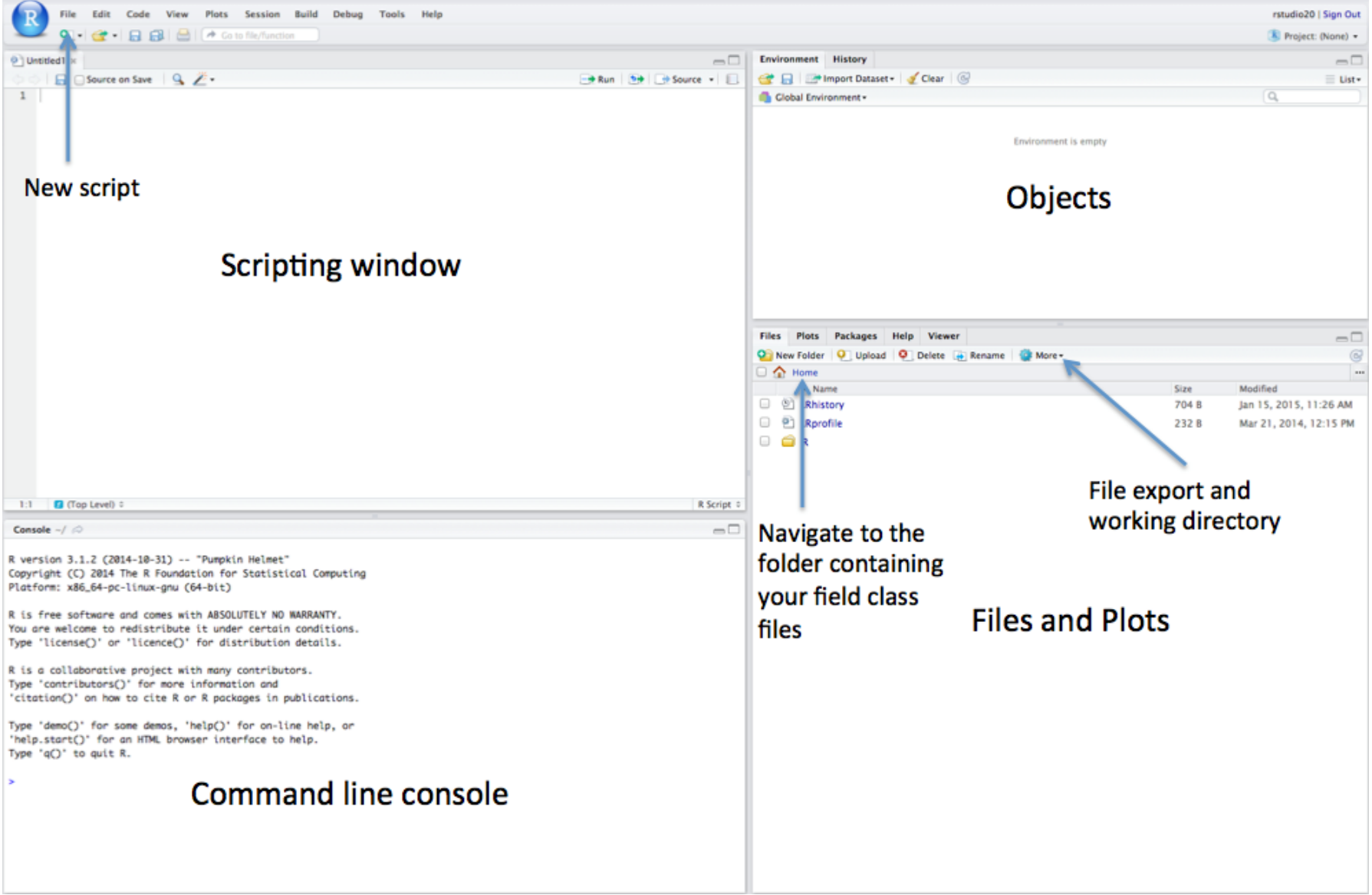
R is a free software environment for statistical computing and graphics. It is extremely powerful and as such is now widely used for academic research as well as in the commercial sector. Unlike software such as Excel or SPSS, the user has to type commands to get it to execute tasks such as loading in a dataset or performing a calculation. The biggest advantage of this approach is that you can build up a document, or script, that provides a record of what you have done, which in turn enables the straightforward repetition of tasks. Graphics can be easily modified and tweaked by making slight changes to the script or by scrolling through past commands and making quick edits. Unfortunately command-line computing can also be off-putting at first. It is easy to make mistakes that aren't always obvious to detect. Nevertheless, there are good reasons to stick with R. These include:

- It's broadly intuitive with a strong focus on publishable-quality graphics. It's 'intelligent' and offers in-built good practice - it tends to stick to statistical conventions and present data in sensible ways.
- It's free, cross-platform, customisable and extendable with a whole swathe of libraries ('add ons') including those for discrete choice, multilevel and longitudinal regression, and mapping, spatial statistics, spatial regression and geostatistics.
- It is well respected and used at the world's largest technology companies (including Google, Microsoft and Facebook), the largest pharmaceutical companies (including Johnson & Johnson, Merck, and Pfizer), and at hundreds of other companies.
- It offers a transferable skill that demonstrates experience in both statistics and computing.

R has a steep learning curve, but the benefits of using it are well worth the effort. Take your time and think through every piece of code you type in. The best way to learn R is to take the basic code provided in tutorials and experiment with changing parameters - such as the colour of points in a graph - to really get "under the hood" of the software. Take lots of notes as you go along and if you are getting really frustrated take a break! To open R click on the start menu and open RStudio. You should see a screen resembling the image below (if it prompts you to update just ignore it for now).

R can be downloaded from https://www.r-project.org/ (https://www.r-project.org/) if it is not on your computer already. Although it is possible to conduct analysis on R directly, you may find it easier to run it via Rstudio which provides a user-friendly graphical user interface. After downloading R, Rstudio can be obtained for free from https://www.rstudio.com/ (https://www.rstudio.com/)

To open R click on the start menu and open RStudio. You should see a screen resembling the image below (if it prompts you to update just ignore it for now).



It is recommended that you enter your commands into the scripting window of RStudio and use this area as your workspace. When you wish to run your commands either hold control Ctrl and enter on your keyboard for each line or select the line you wish to run and click Run at the top of the scripting window.

Our first step is to set the working directory. This is so R knows where to open and save files to. It is recommended that you set the working directory to an appropriate space in your computers work space. In this example, it is the same folder as where the Census data pack has been stored. To set the working directory, go to the **Files** table in the Files and Plots window in RStudio. If you click on this tab you can then navigate to the folder you wish to use. You can then click on the **More** button and then **Set as Working Directory**. You should then see some code similar to the below appear in the command line.

Alternatively, you can type in the address of the working directory manually using the `setwd()` function as demonstrated below. This requires you to type in the full address of where your data are stored.

Lines which commence with hashtags (#) are comments. R will not read these through the console but these are useful for annotating your code for your own benefit.

```
#Set the working directory. The bit between the "" needs to specify the path to th
e folder you wish to use
#you will see my file path below as am example
setwd("C:/Users/Guy/Documents/Teaching/CDRC/Practicals") # Note the single / (\\ w
ill also work).
```

Our next steps are to load the data into R.

Loading data into R

One of R's great strengths is its ability to load in data from almost any file format. Comma Separated Value (CSV) files are often a preferred choice for data due to their small file sizes and simplicity. We are going to open three different datasets from the Census database. Their codes and dataset names are written in the table above. We will be downloading Output Area level data, so only files with "oa11" included in their filenames.

We can read CSVs into R using the `read.csv()` function as demonstrated below. This requires us to identify the file location within our workspace, and also assign an object name for our data in R.

```
# loads a csv, remember to correctly input the file location within your working d
irectory
Ethnicity <- read.csv("camden/Camden/tables/KS201EW_oa11.csv")
Rooms <- read.csv("camden/Camden/tables/KS403EW_oa11.csv")
Qualifications <-read.csv("camden/Camden/tables/KS501EW_oa11.csv")
Employment <-read.csv("camden/Camden/tables/KS601EW_oa11.csv")
```

Viewing data

With the data now loaded into RStudio, they can be observed in the objects window. Alternatively, you can open them with the View function as demonstrated below.

```
# to view the top 1000 cases of a data frame
View(Employment)
```

All functions need a series of arguments to be passed to them in order to work. These arguments are typed within the brackets and typically comprise the name of the object (in the examples above its the DOB) that contains the data followed by some parameters. The exact parameters required are listed in the functions' help files. To find the help file for the function type `?` followed by the function name, for example - `?View`

There are two problems with the data. Firstly, the column headers are still codes and are therefore uninformative. Secondly, the data is split between three different data objects.

First, let's reduce the data. The Key Statistics tables in the CDRC Census data packs contain both counts and percentages. We will be working with the percentages, as the populations of Output Areas are not identical across our sample.

Observing column names

To observe the column names for each dataset we can use a simple `names()` function. It is also possible to work out their order in the columns from observing the results of this function.

```
# view column names of a dataframe
names(Employment)
```

```
##  [1] "GeographyCode" "KS601EW0001"   "KS601EW0002"   "KS601EW0003"
##  [5] "KS601EW0004"   "KS601EW0005"   "KS601EW0006"   "KS601EW0007"
##  [9] "KS601EW0008"   "KS601EW0009"   "KS601EW0010"   "KS601EW0011"
## [13] "KS601EW0012"   "KS601EW0013"   "KS601EW0014"   "KS601EW0015"
## [17] "KS601EW0016"   "KS601EW0017"   "KS601EW0018"   "KS601EW0019"
## [21] "KS601EW0020"   "KS601EW0021"   "KS601EW0022"   "KS601EW0023"
## [25] "KS601EW0024"   "KS601EW0025"   "KS601EW0026"   "KS601EW0027"
## [29] "KS601EW0028"   "KS601EW0029"
```

From using the *variables_description* csv from our data pack, we know the *Economically active: Unemployed* percentage variable is recorded as *KS601EW0019*. This is the 19th column in the Employment dataset.

We will cover more data exploration techniques in the forthcoming practical.

Selecting columns

Next, we will create new data objects which only include the columns we require. The new data objects will be given the same name as the original data, therefore overriding the bigger file in R. Using the *variable_description* csv to lookp the codes, we have isolated only the columns we are interested in. Remember we are downloading percentages, not raw counts.

```
# selecting specific columns only
# note this action overwrites the labels you made for the original data,
# so if you make a mistake you will need to reload the data into R

Ethnicity <- Ethnicity[, c(1, 21)]
Rooms <- Rooms[, c(1, 13)]
Employment <- Employment[, c(1, 20)]
Qualifications <- Qualifications[, c(1, 20)]
```

Renaming column headers

Next we want to change the names of the codes to ease our interpretation. We can do this using the `names()`.

If we wanted to change an individual column name we could follow the approach detailed below. In this example, we tell R that we are interested in setting to the name *Unemployed* to the 2nd column header in the data.

```
# to change an individual column name
names(Employment)[2] <- "Unemployed"
```

However, we want to name both column headers in all of our data. To do this we can enter the following code. The `c()` function allows us to concatenate multiple values within one command.

```
# to change both column names
names(Ethnicity)<- c("OA", "White_British")
names(Rooms)<- c("OA", "Low_Occupancy")
names(Employment)<- c("OA", "Unemployed")
names(Qualifications)<- c("OA", "Qualification")
```

Joining data in R

We next want to combine the data into a single dataset. Joining two data frames together requires a common field, or column, between them. In this case, it is the OA field. In this field each OA has a unique ID (or OA name), this IDs can be used to identify each OA between each of the datasets. In R the `merge()` function joins two datasets together and creates a new object. As we are seeking to join four datasets we need to undertake multiple steps as follows.

```
#1 Merge Ethnicity and Rooms to create a new object called "merged_data_1"
merged_data_1 <- merge(Ethnicity, Rooms, by="OA")

#2 Merge the "merged_data_1" object with Employment to create a new merged data ob
ject
merged_data_2 <- merge(merged_data_1, Employment, by="OA")

#3 Merge the "merged_data_2" object with Qualifications to create a new data objec
t
Census.Data <- merge(merged_data_2, Qualifications, by="OA")

#4 Remove the "merged_data" objects as we won't need them anymore
rm(merged_data_1, merged_data_2)
```

Our newly formed *Census.Data* object contains all four variables.

Exporting Data

You can now save this file to your workspace folder. We will use this data in the forthcoming practicals. Here we are using the `write.csv()` function. We have told the function not to print off the row names (the numbers on the left of the data) as we won't need them. We will save the data as *practical_data.csv* to our working directory. Remember R is case sensitive so take note of when object names are capitalised.

```
# Writes the data to a csv named "practical_data" in your file directory
write.csv(Census.Data, "practical_data.csv", row.names=F)
```

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)