

GEOG0013 Statistical Analysis Worksheet 2: Describing data

Overview

You are now going to use R to prepare and describe some secondary data about 'dwelling' in Barcelona. Importing, preparing and describing data are usually the first and most time consuming steps in a quantitative project. Describing data is not only an important way to start your analysis, it can also help you to spot any obvious problems or errors – for example missing data points or implausible values (e.g. people aged 900). These can wreak havoc with analyses so they need to be identified and addressed early on.

For the rest of this workshop you will be analysing data about the 73 statistical neighbourhoods in Barcelona. A map and a very simple overview of housing and economic issues in the city can be accessed from the Barcelona Field Studies Centre: <https://geographyfieldwork.com/Deprivation-and-Poverty-in-Barcelona.htm>. Read closely but critically through this now.

All of today's datasets have been obtained from the Barcelona Open Data Service (<http://opendata-ajuntament.barcelona.cat/en/>) and are stored on Moodle. This afternoon you have access to four spreadsheets containing information about the (1) average rent (in €), (2) the unemployment rate (in %), (3) the average income (measured as an index where 100 = Barcelona's average) and (4) the percentage of people with foreign citizenship (in %) in each neighbourhood. All data refer to 2015 as this is the latest year available. We are using the *rate* rather than *number* of unemployed and foreign citizens in each neighbourhood as this allows us to compare figures across neighbourhoods with different numbers of people living in them.

Your first task is to import and merge together the four separate files into one R data frame. Once you have done this you will then conduct some basic univariate analysis and graphical plots. Univariate means 'about one variable' – remember that a variable is just something that we measure that can vary (such as the height of students, survey respondents' favourite ice cream flavours, air temperature or finishing position in a race).

As you move through the afternoon you should also reflect on the pros and cons of your secondary data analysis, comparing it to the analysis of primary field data you did in Barcelona and in the last workshop. What types of research questions might be best addressed with each type of data? Can the two types of data complement one another in a single project? What are the practical advantages/disadvantages of each type of data?

Worksheet aims

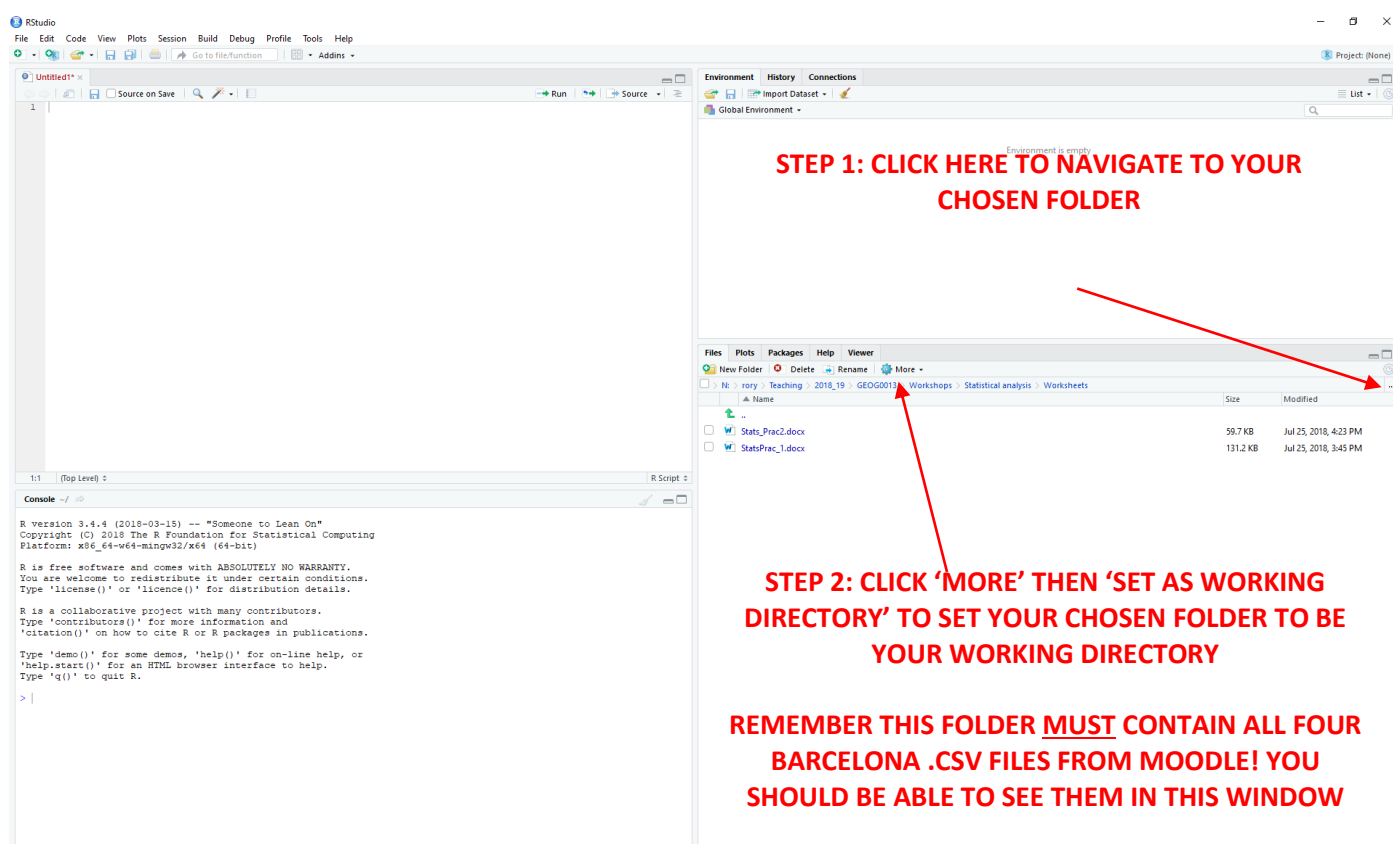
1. Import secondary datasets about Barcelona into R
2. Install an R package
3. Merge dataframes together and rename a variable
4. Learn how to describe variables using plots and summary statistics

Importing data

One of R's strengths is its ability to load in data from almost any file format. Comma Separated Value (CSV) files are our preferred choice. These can be thought of as stripped down Excel spreadsheets. They are extremely simple so they are easily *machine readable*. The four datasets you have been supplied with were all obtained as .csv files from the Barcelona Open Data Service.

Since we are now reading and writing files it is good practice to tell R what the *working directory* is. This is the folder where you wish to store the files you are working with. Before going any further make sure you have downloaded the four Barcelona .csv files you need from Moodle and have these all saved into the folder you are using as your working directory. You will be loading them from here and this is the step where people most commonly go wrong! I suggest creating a new folder on your personal N: drive space to use as the working directory for this workshop.

Once you have the files downloaded into your selected folder, in RStudio, on the lower right of the screen you will see a window with a "Files" tab. If you click on this tab you can then navigate to the folder you wish to use by clicking the little "... " tab on the right hand side. You can then click on the "More" button and then "Set as Working Directory".



The screenshot shows the RStudio interface. The main editor window on the left contains a script with the following text:

```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

The Files pane on the right shows the directory structure: N: > rory > Teaching > 2018_19 > GEOG0019 > Workshops > Statistical analysis > Worksheets. The 'More' button is highlighted with a red arrow. Below the Files pane, the following instructions are provided:

STEP 1: CLICK HERE TO NAVIGATE TO YOUR CHOSEN FOLDER

STEP 2: CLICK 'MORE' THEN 'SET AS WORKING DIRECTORY' TO SET YOUR CHOSEN FOLDER TO BE YOUR WORKING DIRECTORY

REMEMBER THIS FOLDER MUST CONTAIN ALL FOUR BARCELONA .CSV FILES FROM MOODLE! YOU SHOULD BE ABLE TO SEE THEM IN THIS WINDOW

When you click to set your working directory you should see some code appear in the command line. This code can be copied and used to set your working directory in an R script. Remember that you should be writing your code into a script during this workshop so you can save it.

Once the working directory is set up it is possible to load in a csv file using the `read.csv()` function. We are going to load in the four .csv files containing Barcelona data that you have pulled into your working directory from Moodle. Each file's name tells you what information it contains – for example 'Barcelona_rents_2015.csv' contains the data we need on rents, while 'Barcelona_unemp_2015.csv' stores data on unemployment rates. Recall that we need to tell R what to call the data frame objects we are loading in. This is done with the text on the left side of the `<-` sign. So the first line of code below tells R to create a new object called 'Rent' that is a data frame holding the Barcelona_rents_2015.csv file.

```
#Load in the 4 .csv files one by one
Rent <- read.csv("Barcelona_rents_2015.csv") # Data on average rents

Unemp <- read.csv("Barcelona_unemp_2015.csv") # Data on unemployment rates

Income <- read.csv("Barcelona_income_2015.csv") # Data on average incomes

Foreign <- read.csv("Barcelona_fcitizens_2015.csv") # Data on % foreign born
```

You should see 4 data frame objects loaded into R. As you can see in the 'Global Environment' panel, all except the Rent object contains 2 variables (columns) and 73 observations (rows – these rows are Barcelona's neighbourhoods). The Rent data frame has an extra column (variable) that records which of the 10 districts of Barcelona each of the 73 neighbourhoods lies in.

Take a look at the Rent dataset using the `View()` function. Can you see anything amiss?

```
View(Rent)
```

A couple of neighbourhoods have no rent data. R shows an NA value in the rent variable for these neighbourhoods and they will be automatically excluded from most analyses.

Using `print()` will do something similar to `View()`, but this time 'printing' the data into the command window. If you only want to see the top or bottom 10 rows you can use the `head()` and `tail()` functions to print the head and tail of the dataset. This is particularly useful if you have large datasets.

```
head(Rent)
```

```
tail(Rent)
```

To alter the number of rows R displays with these functions you can add an `n=` argument after the command. For example, to display the top 20 rows type:

```
head(Rent, n=20)
```

On your own try using these commands to explore the other dataframes (Foreign, Income and Unemp) you have loaded into R.

Installing a package and renaming a variable

We are now going to install a package called dplyr and use it to rename a variable on the Rent data frame. Dplyr is one part of the powerful 'tidyverse' suite of data science functions for R (<https://www.tidyverse.org/learn/>). You will encounter other tidyverse packages later in the workshop as they are very useful for geographers.

To install an R package from the thousands that are available you need to use the **install.packages()** function, placing the name of the package you wish to install in the brackets. R will then download and install the package from the internet. Use the code below to install dplyr. If a message appears telling you it is already installed then skip this step.

```
install.packages("dplyr")
```

Be patient, adding it may take a little time. You don't need to re-install the package every session as R will automatically save it into your 'library', but each time you open R and wish to use a package you do need to tell R to load it again from the library. You can do this using the **library()** function:

```
library(dplyr)
```

If you encounter an error when trying to use an R function it is often worth checking whether you have loaded the package it comes from out of your library. This is a common cause of problems.

You can now use the **rename()** function supplied with dplyr to rename the 'rent' variable on the Rent data frame to be 'avg_rent'. This makes it easier to understand the variable's contents. Do this now:

```
Rent <- rename(Rent, avg_rent = rent)
```

Within the brackets we first tell R which data frame the variables we are renaming come from (Rent), before then listing the new name we want to assign to our variable followed by its old name (the old name comes after the = sign). Use **View()** or **print()** to take a look at the new Rent data frame – you should see that the rent column has been renamed avg_rent.

Merging dataframes

We can also join, or merge, our four data frames together into a single R object. This is a very useful way to join together datasets from our different sources. Joining two data frames together requires a common field, or column, between them. In this case it is the neighbourhood text field recording the name of each of Barcelona's 73 neighbourhoods in all four of our datasets.

We can merge the four files together in a series of steps. We begin by merging the Foreign data frame (x) onto the Rent data frame (y), telling R that the 'neighbourhood' variable is common to both and should be used to match up the rows of data from the two sources:

```
# Step 1 - merge Foreign onto Rent to create a new file called Barc_neighbs
```

```
Barc_neighbs <- merge(Rent, Foreign, by="neighbourhood")
```

If you inspect the new Barc_neighbs object you will see that data about the % foreign in each neighbourhood has now been merged onto information about that neighbourhood's rent levels.

You are now ready to add Unemp onto the Barc_neighbs data frame in exactly the same way:

```
# Step 2 - then merge in the Unemp data
```

```
Barc_neighbs <- merge(Barc_neighbs, Unemp, by="neighbourhood")
```

Now, on your own, copy and adapt the above code to merge the Income data frame onto the Barc_neighbs object in the same fashion. Make sure you do this before going any further. You can tell you have been successful when you have a data frame called Barc_neighbs that contains 73 rows and 6 variables (neighbourhood, district, avg_rent, p_fcitizen, unemp and income).

Univariate statistics

When describing a variable we usually want to know two things: its *central tendency* and its *dispersion*. Central tendency describes the centre or midpoint of the values. Another way to think about it is as the point around which values tend to cluster. Common measures of central tendency include the mean (the sum of all values on the variable divided by the number of observations – useful for continuous variables that are approximately normally distributed), median (the middle score of a set of ordered observations) and mode (the most commonly observed value – only useful for nominal variables measured as unordered categories).

The `mean()` and `median()` functions can be used to calculate measures of central tendency in R. Do this now for the unemployment rate variable. The code in the brackets uses the \$ symbol to tell R which variable to select from which data frame object. We specify `Barc_neighbs$unemp` to tell R to compute the statistics we want for the unemp variable which is on (\$) the Barc_neighbs object.

```
# Compute mean and then media unemployment rate across Barcelona
```

```
mean(Barc_neighbs$unemp)
```

```
median(Barc_neighbs$unemp)
```

The *dispersion* of values on a variable indicates how spread out they are around the central point. Low dispersion means that the values cluster tightly, whereas high dispersion indicates a wide spreading of values (and hence measures of central tendency may not be very meaningful). Common measures of dispersion include the range (highest value minus lowest value), interquartile range (the value of the 75th percentile observation minus the value of the 25th percentile observation) and the standard deviation (very useful for normally distributed continuous variables and discussed in the Field et al (2012) textbook on your reading list).

Use the code below to calculate the range and standard deviation for the unemployment rate:

```
# Compute range and standard deviation for unemployment rates
range(Barc_neighbs$unemp)

sd(Barc_neighbs$unemp)
```

This is all a bit laborious but fortunately a `summary()` function provides a quick way to get most of this information with one command:

```
summary(Barc_neighbs$unemp)
```

In addition to the minimum, maximum, mean and median you get the value of the 1st and 3rd quartile. These bound the inter-quartile range - more on this shortly.

Histograms and boxplots

Humans are very good at interpreting visual information and so graphical plots are an excellent way to check your data and get a feel for the distribution of values. A histogram is often the best starting point. On a histogram the values of a variable are laid out on the x axis. Columns or 'bins' chop the x axis into segments, with bin height indicating the number (frequency) of cases that fall into each particular segment. Use the code below to create a histogram of the unemployment rate.

```
hist(Barc_neighbs$unemp)
```

To alter the colour of bars add an argument to the function (you can experiment with colouring):

```
hist(Barc_neighbs$unemp, col="green")
```

And then add a title (the 'main =' bit) and label for the x axis...

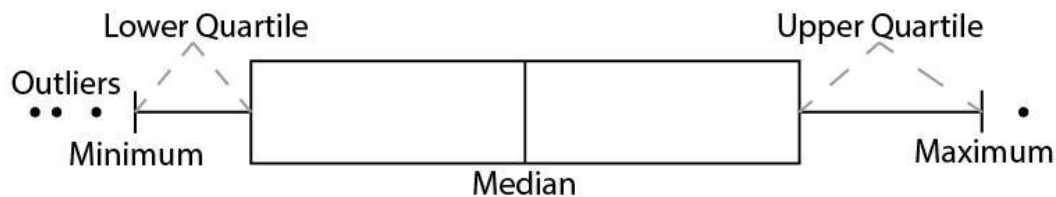
```
hist(Barc_neighbs$unemp, col="green", main="Unemployment rate", xlab="Percentage")
```

On your own try creating a histogram of % foreign citizens in Barcelona's neighbourhoods. To do this copy the code above and alter it. To scroll between plots click on the blue arrow below the file

tab. Hopefully by now you are starting to see why we are training you to analyse data by writing code – it makes reproducing and altering things highly efficient.

We can produce a different type of plot to show some of the summary statistics we obtained in the previous section by using the `boxplot()` function. This will create a box and whisker plot of a variable, in our case the % of neighbourhood residents who are unemployed.

The diagram below illustrates the components of a standard box and whisker plot.



```
boxplot(Barc_neighbs$unemp)
```

We can make the box display horizontally using the `horizontal=TRUE` argument. As you use R you will notice that some arguments (the options written into the brackets) consist of binary TRUE/FALSE options. By default the boxplot command plugs in a '`horizontal=FALSE`' argument, but by specifying `horizontal=TRUE` you can override this to customize the plot.

```
boxplot(Barc_neighbs$unemp, horizontal=TRUE)
```

On your own, try adding a title to the box plot. Hint: you can do this using exactly the same method you used earlier to add a title to a histogram.

A final use of boxplots is to *compare* how the distribution of values on a variable differs across groups of observations. For example, we might be interested in knowing how the unemployment rate in Barcelona neighbourhoods varies across each of the city's 10 districts. To do this we can add the `~` symbol to tell R to draw separate boxplots for groups of cases that are distinguished by the variable listed after the `~` sign. In this case we instruct R to draw separate boxes of the unemp variable for each district in Barcelona.

```
boxplot(Barc_neighbs$unemp ~ Barc_neighbs$district, horizontal=TRUE)
```

You should see that there is a clear geography to neighbourhood unemployment. Which districts have the highest and lowest unemployment in their neighbourhoods? You may need to zoom in on the plot to read all the labels. To do this click the 'zoom' button on the plot window.

Finally, let's save our dataset to the working directory. You can do this using the `write.csv()` function, placing the name of new file into the parentheses after listing the data frame to be saved. Here we just call the new file 'Barcelona':

```
write.csv(Barc_neighbs, "Barcelona.csv")
```

If you finish early try creating neat histograms and boxplots for other variables. Think about the distribution of values on each variable too. Which looks the most/least normally distributed? Which values are the most and least spread out? Remember that you can use the online map from the Field Studies Centre website (reproduced on the next page and at <https://geographyfieldwork.com/Deprivation-and-Poverty-in-Barcelona.htm>) to learn where specific neighbourhoods are in Barcelona. Which seem the most and least affluent areas, and how do your findings compare with the Field Studies Centre description of the socio-economic geography of Barcelona?

You will need to apply some of these methods to complete the worksheet for this afternoon's class so make sure to ask if you don't understand something.

Hints and tips

1. You can save your R script by clicking on the floppy disk icon in the script window. Make sure you do this regularly and always at the end of your session.
2. Trial and error guided by R helpfiles, online searches and textbooks is the best way to learn how to create aesthetically pleasing plots. Try to do some practice on your own in your GEOG0013 independent study time.

BARCELONA

Map of Barcelona showing the number of people per square meter by neighborhood. The map is color-coded into four categories: dark green for areas with more than 149 people per square meter, light green for 100-149, pink for 75-99, and dark red for areas with fewer than 75 people per square meter. Numerous neighborhoods are labeled with their corresponding values.

Neighborhood	Value
Vallvidrera-Tibidabo-les Planes	161
Sant Genís dels Agudells	81
Montbau	76
Clutat Meridiana	51
Vallbona	52
Torre Baró	58
Roquetes	55
Trinitat Nova	47
Trinitat Vella	62
Prosperitat	63
Baró de Viver	38
Bon Pastor	56
La Verneda i La Pau	64
Besòs i Maresme	52
Diagonal Mar i Front Marítim del Poblenou	123
Poblenou	88
Vila Olímpica del Poblenou	149
Parc i la Llacuna del Poblenou	89
Fort Pienc	100
Sagrada Família	96
Camp de l'Arpa del Clot	85
El Clot	81
Provençals del Poblenou	84
Sant Martí de Provençals	68
Navas	78
Sagrera	77
Congrés i els Indians	80
Vilapicina-T. Llobeta	74
Turó de la Peira	65
Porta	66
Guineueta	55
Can Peguera	34
La Clota	79
Vall d'Hebron	92
Teixonera	67
El Coll	114
La Salut	107
Vila de Gràcia	107
Gràcia Nova	98
Dreta de l'Eixample	153
Antiga Esquerra de l'Eixample	121
Nova Esquerra de l'Eixample	108
Sants	85
Hostafrancs	73
La Bordeta	71
Sants-Badal	74
Font de la Guatlla	78
Marina de Port	70
Marina del Prat Vermell-Zona Franca	57
Poble Sec-Parc de Montjuïc	72
Sant Antoni	96
Raval	64
Gòtic	97
St. Pere, Sta. Caterina i la Ribera	89
Barceloneta	76
Les Tres Torres	212
Sarrià	176
Pedralbes	242
Maternitat i Sant Ramon	116
Les Corts	130
Sant Gervasi-Bonanova	193
Penitents-Vallcarca	119
El Putxet i Farró	139
Sant Gervasi-Galvany	189

Source: El Periodico