R Notebook

## Installing R

R is an open-source (i.e. free) alternative to Stata, rapidly growing in popularity. Normally, R is used in conjunction with RStudio, an Integrated Development Environment (IDE) that streamlines the effective use of the software. We are going to install both of these programs now.

R has a steeper learning curve than Stata, but the initial difficulty in learning R is offset by the greater degree of flexibility and capability achievable with the language with more advanced use (i.e. this guide was written and is maintained entirely through R!). Note also that R is in growing demand in both academia and ‘industry’ and will be a highly useful skill for your arsenal.

The first thing you will need to do is install R from <https://cran.r-project.org/> . Under ‘Download and Install R’, select your operating system (e.g. Mac/Windows) and then select ‘base’. This should give you an .exe file to run.

Once you have installed R on your computer, you will want to then download RStudio Desktop from <https://www.rstudio.com/products/rstudio/download> .

If this is your initial time setting up RStudio, I strongly suggest navigating to Tools in the upper toolbar then Modify Keyboard Shortcuts. Find ‘Insert new chunk’, click anywhere in the line, and hold Ctrl-Shift-I - we do this overwriting of the command as the standard keyboard shortcut fails on most UK computers (why? I do not know!). You may also wish to go into Tools, then Global Options, then Appearance to find an R theme that works for you.

## Notebooks & Code Chunks

The top left of the four panes is the Notebook. The majority of your coding will be done here. An R notebook is a document that mixes both text and code, with the code written in ‘chunks’. Chunks are a useful way to conceptualise your code - you write one chunk that accomplishes one subtask. You can run an entire chunk by clicking the green triangle in the top right corner. Alternatively, you can run any line within a chunk by placing your cursor anywhere within the line and pressing Ctrl-Enter. Note that you can have multiple notebooks open at once. You can insert a chunk using the modified Ctrl-Shift-I shortcut

Below the Notebook pane is the Console. When you run code, its output will appear in the console. You can also code directly in the Console. A useful R workflow is to test your code directly in the console, and only put code you know works into the R Notebook chunks. It is also useful for doing quick calculations. The > symbol with a blinking cursor indicates the console is ready to receive code. While code runs, the > symbol will disappear, returning when the code is run.

EXERCISE: Run the first chunk of code in the getting\_started Notebook (below). Where does the output appear?

2+2

## [1] 4

EXERCISE: Calculate 347 divided by 3 in the Console (bottom left) pane. What has R done in the display?

EXERCISE: Insert a chunk that performs the same calculation below this text.

CHALLENGE EXERCISE: Compute the exact area of a circle with radius 9 in the above chunk.

The top right pane contains the Environment pane. In R, like other programming languages, we can assign values to objects. For example, we might want to assign the value of 4 to the letter x. In R we do this with the assignment operator <- . This arrow says take the value on the right, and assign it to the object on the left.

EXERCISE: Run the first line of the chunk below. What changes? Now run the second line. Where does the output appear?

#--- Assign the value 4 to the object x  
x <- 4 #--- First line  
x #--- Second line

## [1] 4

EXERCISE: Add a line to the chunk of code above that assigns the value of 1899 (when LSHTM was founded) to a new object ‘year’. Use R and this new object to calculate how many years it has been since LSHTM was founded.

The bottom right pane contains a number of tabs. We can see the structure of the current working directory under Files, we can see any plots we have generated in Plots, any packages we have installed (more on this shortly) in Packages, and any Help files we might want.

## Packages

Essential for interacting with R, packages are a collection of user-made functions to carry out particular tasks. A function can be as simple as returning a mean of some data, or as complex as processing data and running a statistical algorithm. The key intuition is that functions are collections of code that can execute a particular task. Functions normally take one or more inputs and provide one or more outputs. The collection of functions included in R before loading any packages is called ‘Base R’.

To load the functions in a given package, we first have to install the package. We do this using the install.packages() function. Run the line of code that installs the tidyverse package below by removing the # at the start of the second line to ‘uncomment’ the code. R will install the package to a default directory on your computer. If any dialogue box prompts you to ‘set up a personal library instead’, click yes. You’ll hear a bit more about functions in the session.

#--- Install the package  
# install.packages("tidyverse", dependencies = T)

Once we have the package installed, we must load the functions from this library so we can use them within R. Load the library with the below code.Remember to uncomment by deleting the # symbol from the second line.

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages --------------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1 v purrr 0.3.2  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 0.8.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'readr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts ------------------------------------------------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

The tidyverse contains a number of highly useful functions for visualising, summarising, tidying, and modelling data - the four things we will be doing over the course of these practical workshops. We also need some data to get going with so:

EXERCISE: Install the ‘gapminder’ package and load its library.

Once you have installed both the tidyverse and gapminder packages and loaded their libraries, run the following line of code and then fill in the Google Form - please include your answer to the question of ‘What was the average life expectancy in Europe in 2007?’ Take a second to hypothesise what each line of this code is doing - we will explore these functions in due course.

#--- Find out the average life expectancy in 2007 by continent  
gapminder %>%   
 filter(year == 2007) %>%   
 group\_by(continent) %>%   
 summarise(mean = mean(lifeExp))