Introduction to R for Health Services Research

Reproducible Research

SickKids and DARTH

12/09/2019

This worksheet highlights some key settings where `R` can help make your research easier to reproduce and present to key stakeholder. This session is split into the following sections:

1. R Markdown
2. R Script
3. R Shiny Applications

Throughout the course, we will demonstrate code and leave some empty *code chunks* for you to fill in. We will also provide solutions after the session.

Feel free to modify this document with your own comments and clarifications.

1. R Markdown

Throughout this course, we have provided all materials using R Markdown but we will now briefly highlight how R Markdown can improve the reproducibility of research. In an R Markdown document, you can store, comment on, and run code from **code chunks**. This document can be shared with collaborators or other stakeholders so people can:

- Re-run your code and view the outputs and analysis results.
- Read your free text that explains the analysis and the data.
- Easily modify your code in any way they want.
- Use your document as a template to build their own R Markdown document.

You can **Knit** R Markdown documents to other file types including HTML, PDF and Word. These alternative outputs can be submitted to journals so keep you analysis and manuscript writing in the same document.

- To knit to a HTML document, you will need an Internet browser installed on your computer, but you do not need to connect to the Internet.
- To knit to a Word document, you will need Microsoft Office Word installed on your computer.
- To knit to a PDF document, you will need to install **MiKTex**, an up-to-date implementation of TeX/LaTeX and related programs. You should download the **Net Intaller**, not the Basic Installer. You do not need to *use* LaTeX.

Inline Code Chunks

R Markdown also contains an *inline* feature that allows you to run small pieces of code within the text. This is useful if you want to refer to figures from your data/statistical analysis within the main body of your text, e.g., 82% of trial participants were successfully sedated. Inline code is specified using the following structure: 3. In this case, the **knited** R Markdown document will display the mean of the number 1 to 5 (e.g., 3) in place of the code above. You can **Knit** the document to see.

2. R Script

You can also use `R` with *scripts*. These *scripts* are like a single code chunk from a Markdown document and can be used to write `R` code for more complex operations or to save functions that are reused across multiple documents. `R` scripts can be used to write, store and run `R` code. To open a new `R` script, click

File -> New File -> R Script...

You can type code anywhere in a script. To run a line, you can click on that line and either hit the **Run** button on the top right corner of this window or hit CTRL + ENTER.

EXERCISE 1

- Create a new `R` script.
- Run a few mathematical operations.
- Type in and run `plot(iris$Sepal.Width,iris$Sepal.Length)`. Note that `iris` is a standard dataset that is loaded whenever you open `R`.

Note that when you run code from a `R` script the output *only* displays in the console and the plots display in the bottom right-hand corner of the RStudio interface.

Setting a working directory

The *working directory* in `R` is an important way to ensure that your research results are easy to find and you have loaded the correct dataset. The *working directory* is the folder where all data will be imported from and where all output, scripts and Markdown documents should be stored. You can display the current working directory using the function `getwd()`. You can change the working directory either by clicking **Session -> Set working directory -> Choose Directory** and setting your working directory or writing the command `setwd("path of your working directory")`.

Remember that you should use a forward slash to define your filenames and the path to the folder, e.g. ("C:/").

You can also ask `R` to set the working directory to the folder where you have saved your Markdown document or script using **Session -> Set Working Directory -> To Source File Location**. If you data files are not in the same folder as your `R` script, you have to specify the full path to the data files (as we did in Session 3).

Cleaning the working space and the R memory

To ensure your results are reproducible and not based on previous analysis, it is advisable to begin each Markdown document or `R` script with a command that clears `R`'s memory. This removes all the variables that you used in your previous analysis and any user-defined functions you created. The command to clear you memory is

```
rm(list = ls())
```

You can also clear your R console (your working space) using type CTRL + L.

EXERCISE 2 Create a few variables with different values in the following *code chunk*. Then clear the working space and the R memory.

Adding explanatory comments

It is good practice in any programming language to write code clearly, using logical steps and explaining clearly why those steps were taken. This helps other researchers reproduce your results and review you analysis. It also helps you remember what you did!

In R Markdown, you can explain your analysis using text and add short comments to the code. We have used this method throughout the course. Within an R script, you can only use comments, indicated using the hashtag (#) symbol. Any line with a # symbol will be ignored by R, e.g.

```
# cleaning the memory of R
rm(list = ls ())
```

You can also add comments in the same line as your code to explain what analysis you are doing on each line. For example:

```
rm(list = ls ()) # cleaning the memory of R
```

This commenting approach reduces the lines of code and results in more condensed code but can be more challenging to read if the comments are long.

3. R Shiny Applications

R Shiny is a powerful and elegant package that that allows you to build web-based applications using R. It lets you turn your analyses into interactive web applications without requiring JavaScript, CSS or HTML coding.

To create a new Shiny App, click

File -> New File -> Shiny Web App...

and choose a name for your App and the working directory it will be saved in.

EXERCISE 3 Create a new Shiny App file and name it "demo".

After you create this app, you will see R has opened up a file name `app.R` –> this is the script for your App.

By default, R provides an example of a Shiny App which you can use by clicking the **Run App** button on the top right corner of the Script Editor. This App plots a histogram of the distribution of waiting time to eruption of the geyser from a the Old Faithful Geyser dataset, which is provided by standard in R.

This App interactively generates and displays a histogram with the binwidth the user selects. The user of the App does not have to write or run any R code as everything is done through clicks. This is what makes Shiny Apps cool and powerful!

The following section briefly introduces how to build a shiny app for your analysis. To begin, call the `shiny` library.

```
library(shiny)
```

All shiny apps have two components: **UI** and **Server**.

- **UI** stands for "User Interface", this is where you code what the user can see and click on.

- **Server** contains functions that produce the output that is displayed in or used by the UI.

It is easier to start with the Server since it feeds things into the UI.

In the demo App, the **Server**:

- Contains a function that first stores the 2nd column (eruption wait time) of the Old Faithful Geyser dataset into a variable called `x`.
- Then, it takes different number of bins defined in the UI (called "bins") to create the `bins` variable.
- Finally, it plots the histogram using the appropriate arguments (`x`, `bins`, etc).
- The entire operation is wrapped by Shiny function `renderPlot()` (a function that takes plots produced in `R`) and saved into a variable called `output$distPlot`.
- `output` tells `R` that this is an output to be sent to the UI and `distPlot` is the name given to this operation.

Now we switch to the **UI**,

- This UI has two parts: the side panel and the main panel.
- The side panel contains a side bar with slider input that contains different number of bins (input name is "bins") ranging from 1 to 50, with the default being 30.
- The main panel contains a function called `plotOutput()` that takes our histogram and plots it on the main panel of the App.

To finalize the App, the command `shinyApp(ui = ui, server = server)` is inserted at the very end.

EXERCISE 4 Follow the steps below to build a Shiny App that makes stratified boxplots for the Framingham data:

- Open up the Shiny App file "app.R" in the "framingham" folder.
- You will see that we have selected the categorical variables (the `x`'s) that we wish to stratify our continuous variables (the `y`'s) by.
- `selectInput()` provides a dropdown menu with different user-defined selections.
- Create a dropdown menu for the continuous variables. Include age, sex, current smoking status, diabetic status, prevalent myocardial infarction status and prevalent stroke status.
- Write the code to draw the boxplot under the comment "draw the boxplot of y stratified by x". Make sure you have appropriate labels for the axes and different colours for the boxplots.
- Run the App and test it.

4. Further Reading

There are loads of great resources available to learn more about R:

1. Christopher Gandrud, Reproducible Research with R and RStudio
2. Yihui Xie, Dynamic Documents with R and knitr
3. The tidyverse website, https://www.tidyverse.org/
4. The R Markdown website, https://rmarkdown.rstudio.com/
5. Hadley Wickham, Advanced R.