RStudio tour

Instructor guide

Date revised: 2022-03-02

Notes

In this demonstration, especially at the beginning, it is important that you follow the code instructions exactly. The order and specific actions matter because they introduce specific concepts about how we interact with R, how R interprets commands, and R reacts to commands.

The purpose of the coding demo now is to show only simply **how R code runs** - *not to introduce any functions.*

Preparation

1. Clear your desktop of clutter.

2. Turn off pop-up notifications from desktop versions of WhatsApp, Signal, Teams, etc.

3. Open RStudio and adjust the upper-right R project drop-down menu to close any open projects. This way, when you open RStudio it opens as "Project: None".

4. Open a file explorer window, showing the location where you will save the R project, and have it ready to show.

5. Open this file (or better: print it and have on your desk)

Setup

1. **In Google Meets**:
    - If you have two displays: share your second display, which contains RStudio and the file explorer. The first display will be contain the Meet.

    - If you only have one display, show your entire screen and ask an assistant instructor for help with any questions/hands that you cannot see. (TODO maybe there is a way to avoid mirroring in Meets?)

Icons

1. **Show desktop**
    - Note the desktop RStudio icon, and that there is (or is not) a separate icon for R

Tour of RStudio

1. **R projects**
   - **Adjust the zoom** so the text is readable by the audience. Confirm with one of the assistants.
   - **Open a new R script**.
     - Explain that this is where you will write code commands.
     - Save it as "*demo.R*".
     - Show the file appearing in the **Files pane**, and show it also in the File Explorer.

2. **Tour the RStudio panes:**
   - **Source Pane (R Script)**:
     - R scripts ("instructions") - most commands are run/executed from here
     - Also used to view datasets
     - For Stata-users, its like Do-file and Data Editor windows
   - **R Console Pane**:
     - The R "engine" - note the R version and the "common name" e.g. "Bird Hippie"
     - You can run commands here too (but this is less common)
     - Non-graphic outputs, errors, and warnings
     - For Stata-users, its like the Command Window and also the Results window
   - **Environment Pane**:
     - Shows a list of the datasets you have imported or other objects you have created
     - Briefly note History sub-pane
     - For Stata-users, this is similar to the Variables Manager window
   - **Files, Plots, Packages, Help, and Viewer**
     - Open and rename files
     - View graphic outputs
     - Review your packages (*they won't know what packages are yet*)
     - See help documentation
     - For Stata-users this contains Plots and Project Manger windows

Basic R syntax

1. **R commands**

- Begin typing into the R script
- Type `1 + 2`note that your cursor is in this command line, and **click "Run" with your mouse**.
  - Show that the result appears in the R Console
  - Note the `>` that shows the original command
  - Note the `[1]` that precedes the answer, indicating the first element of an output. This can confuse some people.
- Cut & Paste the same command to a different line in the script. Run this line by clicking "Run".
  - Explain that lines in the command can be run one-by-one, or you can run the entire script line-after-line.
- Add more to the command to make `(1 + 2) * 4 / 6`.
  - Ask the audience what the answer will be, based on order of operations (put in the chat)
  - Run the entire command and see the answer `2`
  - Now highlight just the segment `1 + 2` and click Run.
  - Explain that when some code it highlighted, only the highlighted portion is executed.
- Demonstrate that you can run command by placing your cursor anywhere in the line and typing *Ctrl and Enter* (Cmd and Enter on macs)
- Mention the keyboard shortcut to run commands (Ctrl+Enter, or Cmd+Enter for macs)

2. **Define numeric object**
   - Explain that one reason R is powerful is its ability to save many datasets or values.
   - Explain that you are going to define an object, and discuss R's "assignment operator" `<-`.
   - Explain that it is **like an arrow** and takes the **value** of the right side, and assigns it to the **name** on the left.
   - The assignment operator can be used to **create** an object, or to re-assign its value:
     - In the script, type and run `epiweek <- 34`. Explain that you are creating an object named "epiweek" that has the numeric value 34.
     - Explain that for anyone unfamiliar, an "epiweek" is shorthand for an "epidemiological week" and helps standardise reporting.
     - Show how this new object appears in the **Environment Pane**.
     - Type the command `epiweek` in the R console and see the value `34` appear below
     - Highlight the word "epiweek" *in the R script* and run it. See the value `34` appear in the Console.

3. **Use a defined numeric value**
   - Write the command `epiweek - 1` in the script.
     - Ask the audience what the value will be (enter in the chat).
     - Run the command and note the output `33`.

- Write the command `previous_week <- epiweek - 1`. Ask **"What are we asking R to do?"**
- Briefly note that object names should be short, cannot have spaces, and cannot start with a number
- Ask the audience where changes will appear in RStudio when the command is run.
- Run it, and note Console only shows record of the command - it does not print a value in the Console (note lack of `[1]`). Note new object name and value appears in Environment Pane.
- Note the difference between commands that *print* a object or value, and commands that *define* an object (**"What are you asking R to do?"**)
- Run the command `previous_week` from the script. Note the printed value `33`.

3. **Re-define the value**
   - Below the former commands in the script, *write a new command*: `epiweek <- 35`, and run it.
     - Note the new value of `epiweek` in the **Environment Pane**
     - Run `epiweek` and `previous_week`.
     - Note that `previous_week` is still `33`. Re-run `previous_week <- epiweek - 1` and note the change, and that `previous_week` did not automatically update until the command was re-run.

Commenting

1. **Introduce script header**
   - Explain that now that we have several commands, we should organize our script, for the sakes of ourselves and any future readers.
   - Explain that the "hash" # symbol is used to write freely in the script. Remind people that often the entire script is run at once, not line-by-line, and therefore some areas must be marked by #, so R knows to not try and run it. Otherwise, R will return errors.
   - Begin by writing these few lines at the top, and explain the importance of documentation.

```
# Demo script for intro course
##############################
# Last revised: DATE
# Written by: YOUR NAME
# Contact: contact@appliedepi.org
```

2. **Introduce line-by-line comments**
   - Explain that everyone has their own particular style, but this is one option.
   - Explain that readability is important - give lots of *vertical* space and write clearly.

```r
# Define epiweek
epiweek <- 36

# Define previous epiweek
previous_week <- epiweek - 1
```

3. **Right-side comments**
    - Explain that nothing to the right of the # will be run.
    - Re-write your comments to appear *after* and *to the right* of each command:
    - Use this opportunity to talk about spaces and comment alignment (readability)

```r
epiweek <- 36                  # define epiweek
previous_week <- epiweek - 1   # define previous epiweek
```

Other class of objects

1. Define a character object
    - Introduce the idea of *classes* - but do not introduce `class()` yet, as we have not yet introduced the idea of functions.
    - Note that until now we have dealt with numbers, but you can also define "character values"
    - Write the command `district <- "Bolo"` in the script and run it.
    - Print by running the command `district`.
    - Note the difference between `"34"` and `34` - **note the quotation marks** - one you can use as a number, the other you should not.

Finish

Close RStudio.
Return to slides and begin functions slides.