A few first steps in R

Thies Lindenthal (htl24@cam.ac.uk)

This is by no means a comprehensive R tutorial. More a guide to get far enough to use proper guides.

At any time, search for R tutorials online: A selection of R tutorials.

Getting the software

R can be downloaded here: R

The interface R Studio is available here: R Studio. Select the free version.

After installation, you should see four panes on your screen (which can be re-adjusted later, according to your needs):

1. Console: Here you can execute any command directly
2. Environment: This gives you an overview of any objects that you have created in your session.
3. Source: The pane containing your scripts (you can create new ones with SHIFT-CTRL-N, on Mac: SHIFT-COMMAND-N)
4. Help/Plots: A pane for outputs

Execute code

You can just type any code into the console and hit Enter:

```
1:20
```

Or, better, type the code into a source file or script, highlight the entire line, and run it by hitting CTRL-ENTER (CMD-ENTER on Mac).

Installing additional libraries in R

Sometimes, you wish to extend the core functionality of R and add specialised packages. This is pretty easy - For instance, to add the Stargazer library, just type commands like this into the R Console:

```
install.packages("stargazer")
```

This will install the wonderful Stargazer library. To learn more about a command/function/library, you can always consult the help function by placing a ? in front of a command.

```
?stargazer
?plot
?summary
```

R Markdown is fancy: Combining Code and Text

This document has been written in R Markdown. Check it out here: R Markdown. Installing it might take a while since quite a few bits and pieces of additional software are needed. Might be worth it, though…

```
  install.packages("rmarkdown")
```

Best practices when working empirically

1. Structure your project in a way that allows you to revisit any step of your analysis later. Use scripts. The more steps you automate, the easier it is to fix errors later. Also, it is easier to do sensitiy analysis or robustness checks (changing a few parameters along the way, do everything again?)

2. Keep track of what you are doing. You can add comments to your code by placing a hashtag in front of the line:

```
# This is a comment.
# comments help me remember later what you were doing here and why
# use them extensively!
```

3. Have a clear structure of data inputs, your code, and outputs like tables and figures.

- Folder structures can help: data, code, outputs.
- Data preferably in textfiles like csv: They can be exchanged easily with other software like Excel.
- Some people like working in R Markdown, since it keeps track of code and outputs. Personally, I am not fully convinced, but I can see the merits.

4. Develop a good programming style

- Use functions: Functions help you to avoid producing repetitive code.
- Use meaningful variable names: Call it *price*, not *depvar*. It is difficult enough to keep an overview, don't complicate things unessearily.

5. Make backups regularly, keep distinct versions of your work.

- Versions can just be copies of your "code" folder (quick and dirty).
- If you work in groups, versioning solutions like *git* and platforms like Github might be a good idea.