Quant II

Lab 0: Intro to R Markdown, R, and reproducibility

Giacomo Lemoli

January 31, 2022

Ciao!

- Giacomo Lemoli, 4^{th} year PhD.
- I do research on identity formation in CP and (H)PE
- Fond of: (i) the politics of language and culture, (ii) stats and causal inference
- Personal website: giacomolemoli.com
- Email: gl1759@nyu.edu
- Office: 404

Logistics (1): recitations

- Lab: Mondays, 2pm - 4pm EST, Room 435
- Lab materials will be posted on the lab's Github repo
- For class material refer to Course web page
- Office hours: by appointment

Logistics (2): homeworks

- Homework due via email to Cyrus and me by 5pm ET on the indicated day
- Submit **PDF document** + **Code used**
    - Markdown recommended (more below)
    - Alternatively, PDF and separate R/Stata code
- Code should be well commented (more below)
- Tables and plots format should be of high quality, irrespective of the editor used
- Relevant information of the answer should be presented in the written paragraphs
- Ultimately, the goal is to learn how to **do** and **communicate** empirical research

Lab resources

- Extensions of concepts seen in class
- Code samples and packages useful for homeworks
  - R (primary) and Stata (secondary)
  - In line with the department/discipline norms, we focus on R
  - Stata resources will be posted as homeworks require to work with Stata-written replication material

Today's plan

- R Markdown
- Basic R and examples
- Extra Basic R (time allows)
- Doing reproducible work

Install R

- It depends on whether you are a Mac or Windows user
- For MAC, go to http://cran.r-project.org/bin/macosx/
- For Windows, go to http://cran.r-project.org/bin/windows/base/
- You should choose an editor and learn how it works
    - One common choice is RStudio, but there is vim, emacs, Notepad++, etc.
- If you are Python/Data science people, you can also download and run RStudio from Anaconda

When things break

- Documentation - E.g.: `?lm` (equivalent of Stata's `help regress`)
- Google
- CRAN (Reference manuals, vignettes, etc) - E.g.: http://cran.r-project.org/web/packages/AER/index.html
- JSS - E.g.: http://www.jstatsoft.org/v27/i02
- Stack Overflow - http://stackoverflow.com/questions/tagged/r
- Listservs - http://www.r-project.org/mail.html

Resources

- The Art of R Programming - N. Matloff
- Modern Applied Statistics with S - W. Venables and B. Ripley
- Advanced R Programming - H. Wickham
- The R Inferno - P. Burns
- Rdataviz - a talk by P. Barberá on ggplot2
- Basic Intro to R - also by P. Barberá
- Jamie Monogan

Resources

- Harvard Dataverse and other replication directories for scientific work
- Reproduction/Replication is the best way to learn how to work with data
- I encourage you to always look at the code and data of papers which use techniques you are interested in

What is R Markdown?

- A tool within RStudio that allows you to write documents, presentations, or webpages, and combine written text with code
- The text in the document can be fully formatted
- You can choose whether to make your code visible or not
- Output documents can be PDFs, Word Documents, HTML pages, and other formats

Why R Markdown?

- It has all the advantages of LaTeX for writing and formatting scientific documents
- Can directly embed code **and** code output in the document
  - Everything (data analysis, graphs, tables, text) is in one place
- Work is directly reproducible and verifiable
  - Besides Quant 2 homeworks, I recommend to use it for all class projects that involve replications or data analyses

What if I prefer Stata?

- You can still use Markdown via Stata Markdown
  - Similar to R Markdown, you can write from your Stata editor
- Up-to-date Stata software is available on NYU desk machines

Getting started with R Markdown

- You need to download R and RStudio. For recent versions, that's enough
- You may need to install some packages the first time (`knitr`,`markdown`)
- If you want to generate PDF output, you also need a LaTex system (e.g. MiKTeX)

Getting started with R Markdown

Once you have done all that,

1. Open RStudio
2. Select File > New File > R Markdown
3. Enter Title, Author, Output Format (which can easily be changed later)
4. You are good to go! You will see that there is some example text that you can delete.

Markdown document

The documents typically have four different pieces: the **YAML**, the **formatted text**, **code chunks**, and **inline code**.

YAML

- At the beginning of any R Markdown script is a YAML header section enclosed by —.
- Includes title, author, date, and type of document you want to produce -`beamer_presentation`, `pdf_document`, `html_document`, etc
  - An overview of the available output formats is here
- **NB**: rules in the header section will alter the whole document

You will always insert something like this at the top of your new .Rmd script:

```
---
title: "Quant II"
subtitle: "Lab 0: Intro to R Markdown, R, and reproducibility"
author: "Giacomo Lemoli"
output: pdf_document
---
```

Code chunks

- The real payoff of R Markdown
- The code you enter is executed and the results are displayed in the document
- The syntax for code chunks looks like this:

```
'''{r}
R code here
'''
```

Code chunks

- Note that “‘ are backticks, not quotes or apostrophes. Everything that goes between these backticks should have R syntax

```
set.seed(1)

x <- runif(10)
y <- rnorm(10, x, 1)
df <- data.frame(x, y)
df
```

```
##             x           y
## 1  0.26550866 -0.55495972
## 2  0.37212390  0.85955295
## 3  0.57285336  1.31117807
## 4  0.90820779  1.48398914
## 5  0.20168193 -0.10370646
## 6  0.89838968  2.41017085
## 7  0.94467527  1.33451851
## 8  0.66079779  0.03955721
## 9  0.62911404 -1.58558584
## 10 0.06178627  1.18671719
```

Code chunk options

- You can specify options for each code chunk

```
' ' ' {r name, echo = TRUE, eval=TRUE, warning=FALSE, message=
R code here
' ' '
```

- name - A label for your code chunk
- echo - Whether to display the code chunk or just show the results. If you don't want the reader of the document to see the code, but just the output, you can set echo=FALSE
- eval - Whether to run the code in the code chunk
- warning - Whether to display warning messages in the document
- message - Whether to display code messages in the document

Output visualization

```r
iris <- read.csv("iris.csv", sep = ",")

head(iris)
```

```
##   X Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1 1          5.1         3.5          1.4         0.2  setosa
## 2 2          4.9         3.0          1.4         0.2  setosa
## 3 3          4.7         3.2          1.3         0.2  setosa
## 4 4          4.6         3.1          1.5         0.2  setosa
## 5 5          5.0         3.6          1.4         0.2  setosa
## 6 6          5.4         3.9          1.7         0.4  setosa
```

Output visualization

```r
fit <- lm(Sepal.Length ~ Sepal.Width, iris)

summary(fit)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5561 -0.6333 -0.1120  0.5579  2.2226
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.5262     0.4789   13.63   <2e-16 ***
## Sepal.Width  -0.2234     0.1551   -1.44    0.152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8251 on 148 degrees of freedom
## Multiple R-squared:  0.01382,    Adjusted R-squared:  0.007159
## F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

Output visualization

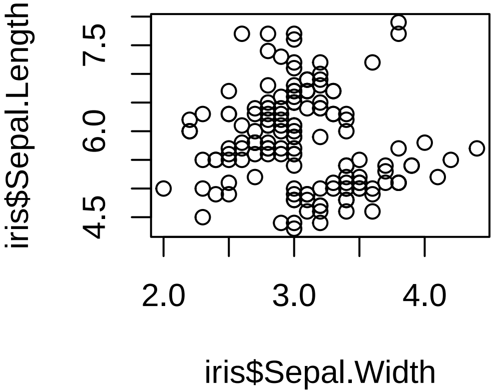
```r
library(stargazer)

stargazer(fit, type = "text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                            Sepal.Length
## -----------------------------------------------
## Sepal.Width                   -0.223
##                               (0.155)
##
## Constant                     6.526***
##                               (0.479)
##
## -----------------------------------------------
## Observations                    150
## R2                             0.014
## Adjusted R2                    0.007
## Residual Std. Error      0.825 (df = 148)
## F Statistic            2.074 (df = 1; 148)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

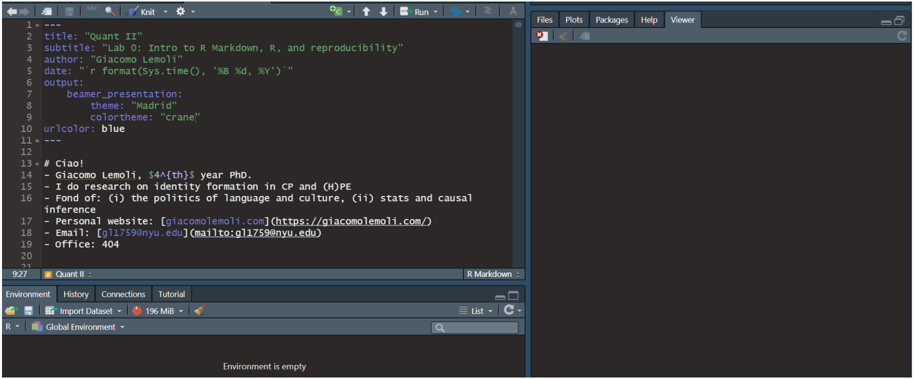
Output visualization

```
plot(iris$Sepal.Width, iris$Sepal.Length)
```



Output visualization

`![](screenparts.png)`



```
---
title: "Quant II"
subtitle: "Lab 0: Intro to R Markdown, R, and reproducibility"
author: "Giacomo Lemoli"
date: "`r format(Sys.time(), '%B %d, %Y')`"
output:
    beamer_presentation:
        theme: "Madrid"
        colortheme: "crane"
urlcolor: blue
---

# Ciao!
- Giacomo Lemoli, $4^{th}$ year PhD.
- I do research on identity formation in CP and (H)PE
- Fond of: (i) the politics of language and culture, (ii) stats and causal inference
- Personal website: [giacomolemoli.com](https://giacomolemoli.com/)
- Email: [gl1759@nyu.edu](mailto:gl1759@nyu.edu)
- Office: 404

```

Formatted text

It is very easy to format your text in Markdown.

- `*italics*` *italics*
- `**bold**` **bold**
- `~~strikethrough~~` ~~strikethrough~~
- `[nyu](https://www.nyu.edu/)` nyu
- `superscript^2^` superscript^2

You can find many more here, including headers, lists, etc.

Or Google your way out of it, as for everything

Equations

- Use LaTex notation
- You can include them either between single dollar signs (inline equations) or double dollar signs (displayed equations)

We can use inline equations such as `$y_i = \alpha + \beta x_i + e_i$` which is displayed as $y_i = \alpha + \beta x_i + e_i$.

Or displayed formulas:

```
$$f(x) = \frac{e^{(-x-\mu)^{2}/(2\sigma^{2})}}
{\sigma \sqrt{2\pi}}$$
```

Which looks like this:

$$f(x) = \frac{e^{(-x-\mu)^{2}/(2\sigma^{2})}}{\sigma \sqrt{2\pi}}$$

Equations

For multiline equations, you need the `aligned` environment:

```
$$
\begin{aligned}
x & = 2*z + 3 \\
y & = 5*z + 6
\end{aligned}
$$
```

$$\begin{aligned} x &= 2 * z + 3 \\ y &= 5 * z + 6 \end{aligned}$$

Inline code

- You can embed R output within text
- Use [r R-object] within the single backticks

```
mean <- mean(x)
```

The mean of x is bt-r mean-bt

The mean of x is 0.5515139

- Text gets updated automatically when code is updated too

Reproducibility

- One of our learning objectives in this class (and in other classes)
- Making sure that our research is reproducible is a goal to maintain while working
- It should be as easy as possible to reconstruct all the steps that go from data to results
    - First and foremost for us and co-authors/collaborators, then for everyone else
- For empirical research: structure of data files and folders, coding style

Files/folders

- Files should be set-up and organized in a coherent way
  - E.g. not keeping everything in the Desktop folder
- Unique directory for the class
- Inside it, new directory for each new project
- Within the project: use sub-directories
  - E.g. a “Raw data” folder to store data, a “Codes” folder for the scripts, an “Output” folder for saving the results
- From inside the script you can use the project folder as main directory
  - `setwd("path_to_project_dir")` or `cd "path_to_project_dir"`
- Then your code pulls files from and saves files to the specific sub-directories

Code

- Coding style is like writing style: as personal as it can be, but should follow guidelines
  - Key principles are efficiency, clarity and transparency (mostly for ourselves and co-authors/collaborators)
- Code script should be concise and as short as possible
- Exhaustive and constant in-line comments
  - Every line (or sequence of lines that performs a self-contained task) should be preceded by a comment explaining what it does
- Avoid redundancy
  - If you need to execute the same action multiple times, it is better to write a function for that action and then just execute it when needed (or use loops)
  - Copy-and-paste of code chunks makes errors more likely and is hard to read through
- Play around with replication packages to learn coding tricks

Resources

- Great resource: Gentzkow and Shapiro (2014), Code and Data for the Social Sciences: a Practitioner's Guide