Studio 2: Introduction to R

CS100: Studio 2

Introduction to R

September 22, 2021

Instructions

During this week's studio, you will be learning how to use the dplyr library, and to produce plots using the mosaic library. You will be writing your code in R Markdown, and you will also be using RStudio to interface with R.

So far, you have been conducting most of your data analyses using spreadsheets. While R is a different and more powerful tool, the concepts you have learned will continue to apply.

Upon completion of all tasks, a TA will give you credit for today's studio. If you do not manage to complete all the assigned work during the studio period, do not worry. You can continue to work on this assignment until Tuesday, September 28, at 10 PM. Come by TA hours any time before then to show us your completed work and get credit for today's studio.

Objectives

By the end of this studio, you will be able to use R to conduct a basic data analysis and produce basic visualizations. Specifically, you will learn about:

- Libraries
- R Markdown
- Data analysis in `dplyr`
- Visualizations using `mosaic`

Tips

In this course (and beyond, hopefully), you will be using the R programming language to—you guessed it!—write (computer) *programs*. Computer programs are chunks of code that people write to automate tasks. Data analysis is an example of a task that it can be very useful to automate. You may think, at first, that you need only analyze data once, so that automation is overkill. But as you engage more and more in data science, you will discover that your initial analyses can almost always be improved, making it much more efficient for you to automate as much of the analysis process as possible. Moreover, other data scientists (including your later self) can better understand your methodology and/or potentially reproduce your results when you record the steps of your analysis in a computer program.

In RStudio, programs are written in the scripting window. But before you write any code in the scripting window, you can – and should! – debug it in the console. A very good rule of thumb (even for seasoned programmers) is to run *each and every* line of code *one at a time* in the console, to check that it produces the expected results. Then and only then should it be copied into a script. The staff always write their code in the console first, and only copy it to an R script or R Markdown file after verifying that it works.

Before any data analysis can be performed, the data almost always needs to be prepared, or **cleaned**. You may imagine that you will clean the data only once, and then you will be ready for your analysis, so that this cleaning can be done in the console. Like your analysis code, which we already argued is is worth saving so that it can be tweaked and then repeated, and/or scrutinized by others, your cleaning code should likewise be saved. Feel free to

clean your data in the console; but be sure to save all your cleaning commands in an R script or R Markdown file, so that you can rinse and repeat, as necessary.

Setup

Before proceeding with the studio, you will need to install a few R libraries. They are: `dplyr`, `ggplot2`, `mosaic`, and `manipulate`. Open RStudio, and then run the following commands in the console:

```
install.packages("dplyr")
install.packages("ggplot2")
install.packages("mosaic")
install.packages("manipulate")
```

R Markdown

A **markup** language is a language which uses tags within documents to indicate how the various components of the document should be rendered. **HTML**, the language in which web pages are written, stands for Hypertext Markup Language, because text can be "marked" with links to another web page, making them *hypertext* instead of plain ol' text. This document was written in **Markdown**, which is a very lightweight (i.e., easy to use) markup language—hence, the clever name Mark*down*.

In CS 100, you will create documents using an extension of Markdown called **R Markdown**, which enables the embedding of R code and R plots in Markdown files. Starting in today's studio, you learn to use this tool to present your data analyses to others, by writing an *R* Markdown file, converting it to HTML, and posting it on the course website.

Create a new R Markdown file by going to `File → New File → R Markdown...`. If this is your first time creating an R Markdown file, R studio will prompt you to install a few packages. Allow the process to proceed. Upon completion, you will see a window in which to name your file, which looks like this:

Set Title to `CS100 Studio 2` and Author to your name(s). Leave `Document` selected on the left and `HTML` selected under the "Default Output Format". Once you click `OK`, the file will be created with a default template filled in. Read through the content of the R Markdown template. When you are ready, click the arrow next to `Knit` in the toolbar of the RStudio scripts pane to see how the document renders as HTML.

Most, if not all, of the basic features of general Markdown files are also available in R Markdown. What is unique is the ability to also add snippets of R code and their outputs. Together, these are called "R code chunks".

An example of an R code chunk is:

````
```{r Code Chunk}
x <- 0
x
```
````

Note that the part in curly brackets must start with 'r'. In other words, it should look like {r Chunk Title}.

When you render this R Markdown code, `0` will be assigned to `x`, and the value of x, namely 0, will be displayed.

If you want to hide your R code, but display its output, you can add `echo = FALSE`:

````
```{r, echo = FALSE}
x <- 0
x
```
````

In this example, the value of x will be displayed, but the underlying code will not be rendered. You should use `echo = FALSE` whenever you want to depict a plot, but hide the code that generates it.

Alternatively, you may wish to display code without running it (or displaying its output). To accomplish this, use `eval = FALSE`:

````
```{r, eval = FALSE}
x <- 0
````
````

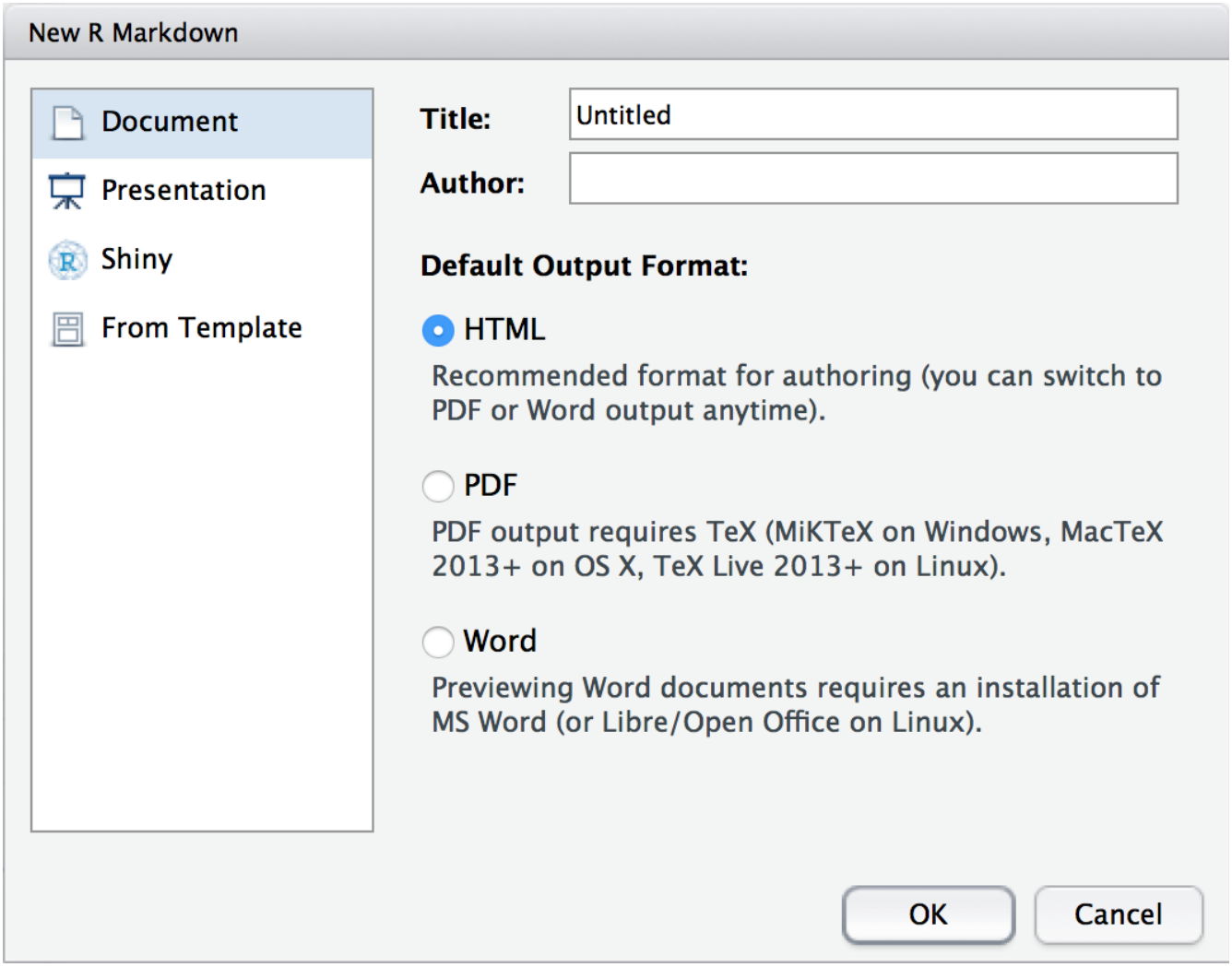

Figure 1: new_file

```
x
```
```

**Note:** In this studio—and in most, if not all assignments—you should not use `echo = FALSE` or `eval = FALSE` within your code chunks, because we can more easily grade your work when both your code and your output are in the same place. (Likewise, other data scientists critiquing an analysis benefit from seeing both the code and the output in the same place; hence, the utility of RMarkdown as a tool for presenting data analyses.)

Some more details on R code chunks can be found here. In addition, here is a handy cheat sheet for R Markdown.

In today's studio, you will be writing a report in R Markdown. At any time during this studio, you should feel free to click `Knit HTML` to see how your R Markdown file renders.

Before proceeding, delete the lines after the header in the default template (leave only the title and output section), and save the file.

### Data

In 2013, students in a Slovakian statistics class were asked to administer a survey to their friends, ages 15 through 30. The survey consisted of 150 questions and was completed by 1010 individuals. Of these variables, 139 are integer-valued, and 11 are categorical. In addition, missing values are present in the data as 'NA'.

If you refer back to lecture 6b, you will find a quick introduction to dplyr using the Slovakian youth survey data set. (Short descriptions of the questions are also listed in this file.) In lecture, we formulated a variety of hypotheses based on very incomplete pictures of the data—only the first 6 rows of various subsets of the data. Today in studio, you will do better: you will query the data set, and then you will analyze the results of your queries in their entirety.

### Setup

To get started, add the following code chunk to your R markdown file, and then run it by clicking 'Run' and selecting 'Run All'. Make sure to include the three apostrophes before and after the code, as this tells R where the code chunk begins and ends.

```
```{r setup, include = FALSE}
library(dplyr)
library(ggplot2)
library(mosaic)
library(manipulate)

responses <- read.csv("http://cs.brown.edu/courses/cs100/studios/data/2/responses.csv", sep = ";", header =
```
```

This code loads the `dplyr` library, which we will use for data wrangling, and the `ggplot2`, `mosaic`, and `manipulate` libraries, which we will use for plotting. It then reads in the data.

*Important Note:* The `install.packages` command should only ever be typed into the Console; it should not be part of any scripts or R markdown files. This command downloads packages from the web and then installs them on your machine, which need be done only once. A series of `library` commands, on the other hand, should be included at the top of any script/R markdown file that makes use of the respective libraries. The `library` command loads a previously-installed package into the current environment for present use.

At this point, we are ready to take a look at a few of the hypotheses mentioned in lecture.

**Hypothesis 1: Does having more friends increase one's happiness?** To investigate this first hypothesis, we'd like to determine whether or not there is a relationship between the respondents' number of friends and their happiness levels.

Create a new chunk of code. From `responses`, select `Number.of.friends` and `Happiness.in.life`, sort by `Number.of.friends` in descending order, and save the output to the variable `friends_vs_happiness`. If you are having trouble, refer back to the lecture notes, which include the necessary functions. When done correctly, the top 6 rows (`head(friends_vs_happiness)`) should look like:
```

```
  Number.of.friends Happiness.in.life
1                 5                 5
2                 5                 5
3                 5                 4
4                 5                 4
5                 5                 4
6                 5                 2
```

In your R Markdown file, create a new section titled "Hypothesis 1", and write your code for this problem in a code chunk. You can add simple text, such as titles and descriptions, to an R Markdown file outside the R code chunks, which are enclosed by three apostrophes. Adding '#' symbols in front of a line of text increases that line's font size in the output file.

Next, we will plot these two variables against one another. Using the `mosaic` library, we can plot `friends_vs_happiness` with `mplot` in the Console:

```
mplot(friends_vs_happiness)
```

You will receive the following prompt in the console:

```
Choose a plot type.

1: 1-variable (histogram, density plot, etc.)
2: 2-variable (scatter, boxplot, etc.)
3: map
```

Let's plot our two quantitative variables using a scatter plot. To do so, type in `2` and then enter.

The plot should appear in the bottom right. When you see the plot, click on the gear symbol in the top left. A *manipulate* window with sliders and drop down menus will appear. If it doesn't, make sure the 'Plots' tab is open (not the 'Files' tab or the 'Packages' tab, etc.), and try running the command in the console instead of in the R Markdown window.

For the graphics system, make sure to choose `ggplot2`. For this hypothesis, select `jitter` as the "Type of plot". Jitter adds a small random value to each point. Without jitter, you would not see all the responses, because their exact values overlap. Also, switch the variables on the two axes, so that "Number.of.friends" is the explanatory variable and "Happiness.in.life" is the response variable, rather than the other way 'round.

To see the R code that generated your plot, click on the `Show Expression` button. Copy and paste the expression into your R markdown file within its own code block.

Next, let's add color to the plot according to "Number.of.friends". To do this, select "Number.of.friends" as the "Color" in the manipulate window. As above, click `Show Expression`, and then copy and paste this new expression into your R markdown file in another code block.

If you want to manipulate your plot further, go ahead. As you do so, click on `Show Expression` to see how the underlying R code changes.

To exit the manipulate window, click on the `>>` in the upper right-hand corner.

Does there seem to be a relationship between the number of friends someone has and their happiness?

Some calculations might help. Let's average "Happiness.in.life" by "Number.of.friends". To do this, group the data by "Number.of.friends", and then summarize the average "Happiness.in.life".

How did that work out? Not too well, we are guessing. The problem is there are "NA"s in the data, and the `mean` function cannot function (ha ha!) when "NA"s are present. The fix for this is easy: simply add the argument `na.rm = TRUE` to the `mean` function.

Now does there seem to be a relationship between the number of friends someone has and their happiness? Explain your answer in your R markdown file.

Hypothesis 2: Does gender relate to happiness among Slovakian youth? Now we'd like to determine if there is a relationship between gender and happiness. Using `dplyr`, select "Gender" and "Happiness.in.life", and omit NAs. Assign the output to `gender_vs_happiness`. The top 6 rows (`head(gender_vs_happiness)`) should be:

```
  Gender Happiness.in.life
1 female                 4
2 female                 4
3 female                 4
4 female                 2
5 female                 3
6    male                 3
```

We will use a boxplot to summarize the range of values for happiness by gender. Like before, type in the console:

```
mplot(gender_vs_happiness)
```

But this time select a boxplot, still using the `ggplot2` graphics system.

Again, it's not all that easy to see what's going on from this plot. Instead, it could be better to calculate the average "Happiness.in.life" by gender. Following the pattern above, group the data by "Gender", and then summarize the average "Happiness.in.life". (Don't forget to remove "NA"s.)

In your R markdown file, create a section titled "Hypothesis 2", and insert your `dplyr` code and the boxplot expression (which again you can obtain by clicking on `Show Expression`) in a code chunk under this title.

Which youth were happier in Slovakia in 2013: males or females? Describe your findings as they relate to the hypothesis.

Hypothesis 3: What about happiness and age? Finally, let's investigate whether there is a relationship between happiness and age. Start by simply plotting "Age" vs. "Happiness.in.life", adding jitter to the response variable.

There does seem like there *might* be a relationship lurking, but the plot is a bit busy. Instead of plotting *all* the points, let's group by "Age" and compute averages once again (removing NAs), and then plot average "Age" vs. "Happiness.in.life".

To do so, after grouping by "Age" and computing averages, assign the data frame that results to a new variable called `happiness_vs_age`. Then, use mplot() to generate a scatter plot of `happiness_vs_age`, with the `x` variable set to "Age" and the `y` variable set to average happiness.

Include all your code for this hypothesis in code chunks under a section entitled "Hypothesis 3". Report whether you observe a relationship between the two variables.

Conduct your own analysis

To conclude this studio, we would like you to explore the data set a bit more on your own to come up with your own hypotheses. To see all the variables in the data set, you can use the R command `names(responses)`, and to better understand what each variable represents, you can read through the survey questionnaire.

Investigate 3 more hypotheses (so that you have a total of 6). For each hypothesis, you should:

- Use `select()`, `filter()`, `arrange()`, `mutate()`, `group_by()`, or `summarize()` to manipulate the data
- Visualize the data with charts produced with `mplot()`
- Describe your findings

Include all of your code and results in your R Markdown file.

End of Studio

When you are done please call over a TA to review your work, and check you off for this studio. If you do not finish within the two hour studio period, remember to come to TA office hours to get checked off.