Instructions

Intro to R: Data Frames, Lists, Tidyverse, and Intro Plotting

ICPSR: Data Visualization (Prof. Cooper)

Week 1: Introduction to R

- Date: June 25th - 29th
- Time: Due June 29th, 9:00 PM to Canvas

Preparation

1. In addition to having watched my videos and Datacamp videos, you should have...
    - installed R,
    - installed RStudio, and
    - set RStudio's Global Options. It is recommended that you deselect "Restore `.RData` into workspace at startup" and set "Save workspace to `.RData` on exit" to "Never".
2. During this term, you should consider using a specific folder within your computer – your "working directory" – where you will save your R scripts, data, figures, and other documents. To make this folder, open RStudio and go to "File" > "New Project". Then, choose "New Directory", and make a new "project" (i.e., a new directory) somewhere on your computer. The project name, which is the name of your working directory, can be anything. Check that the new folder is generated with a new `.Rproj` file. As long as all of your files for this course are saved in this working directory, you need to make this `.Rproj` file once. When you work on your own project after this course, however, you should specify a different working directory with a new `.Rproj` file for each project.

What should you do in Week 1?

- In RStudio, choose "File" > "New File" > "R Script". Then, save the file as "Week01_script.R".

- Work on Exercises below.
- You should upload your R script by June 29th, 9:00 pm to Canvas ("Assignments" > "Week 1 Assignment").

Notes:

- Normally I would ask you to not write or type your answers *within your script*. But feel free to write some comments (with `#`) for your own memo. At this stage I encourage you to write reasonably extensive comments in your own version of your scripts so that you remember what it is you did and why you did it. Scripts to be read by others, under normal circumstances, should not be quite so heavyhanded with comments.
- Never use `T` and `F` as synonyms for `TRUE` and `FALSE`, because `T` and `F` can be redefined and could cause confusion and error. See `https://www.r-bloggers.com/r-tip-avoid-using-t-and-f-as-synonyms-for-true-and-false/`
- Each time I ask you to create a new object, please follow the creation of that object by printing the object. Under normal circumstances you would not print out every object in your R script, whether using proper workflow on your own or sharing scripts with other researchers and collaborators. It will be useful for all of us *in the context of this course*. You can check your work constantly by seeing the output and I can grade more efficiently by seeing the output.
- If I use a particular verb or reference a particular operator or function that you do not immediately recognize, there are many resources out there to help you to discover the solution. The process of search and discovery for basic coding solutions is a part of programming, and **R** has many online resources for you. Google is your friend, as is any one of a number of sources for **R** support.

Exercises

Set up the top of your R script with your name, the name of the course, the term, and the date using comments. There are different style guides for comments. For your homeworks, please use `#` (followed by one space) to add a comment to your code. Begin with `##` (followed by one space) for entirely commented lines. **NOTE**: the latest version of the World Income Inequality Database (WIID) has strangely coded region variables (different from the prior 2018 version, which made more sense).

```
> ## Data Visualization (ICPSR) Summer 2021
> ## Intro to R Part 2 \& Intro to ggplot2
> ## Name: YOUR NAME GOES HERE
> ## Date: June 29th, 2021
```

Factors, Data Frames, and Lists

1. **Categorical Variables: Characters & Factors**

   (a) Create an object called `truth` that takes the value "ICPSR is the best summer stats program!"

   (b) Check the `class` of `truth` and print/report it.

   (c) Create a vector called `colors` that takes the following values: red, blue, green, red, and blue. Put the values in quotations inside the vector.

   (d) Create a factor vector called `factor_colors` from the `colors` vector.

   (e) Create another factor called `factor2_colors` from the `colors` vector. This time set the argument `levels` to just red and blue. The result should be different from the previous factor vector.

      i. What does this tell you about factors? Answer briefly in a comment.

      ii. You are going to need to know about the properties of factors in this class and pretty much all your future research!

   (f) Report summaries of `colors`, `factor_colors`, and `factor2_colors`.

      i. Note the differences between the summaries of characters and factors. Note also the differences between two factors of the same variable but with different levels.

   (g) Create a vector called `ideology` that take the following 10 values: "liberal," "conservative," "very liberal," "very conservative," "middle of the road," "slightly conservative," "slightly liberal," "liberal", "conservative," "middle of the road."

   (h) Now create a factor called `fact_ideo` from the `ideology` vector. Make sure that the levels of `fact_ideo` are ordered appropriately, from "very liberal" to "very conservative." There is more than one way to reorder a factor.

(i) Create a vector called `respondent` that takes the following 10 values: Susie, Abdul, Maria, Fred, Wilma, Barney, Dino, Ajax, Thor, and Betty. Put them all in quotations in the vector. This will be used later.

2. **Data Frames**

(a) Create a vector called `income` that takes the following values: 100000, 75000, 48000, 62000, 31000, 52500, 274000, 88000, 21000, and 74000. Don't add commas inside the numerical values; **R** will incorrectly read the values as characters if you do.

(b) Create a data frame from your `ideology`, `respondent`, and `income` vectors/variables, calling it `data1`.

(c) Show me the head of `data1`.

(d) Show me the tail of `data1`.

(e) Show me structure of `data1`.

(f) Order the observations in the dataset by `income` in descending order (highest first), making a new copy of the data and calling it `orderdat`.

(g) Go back to `data1` and extract from `data1` the respondent with the lowest income; subset/isolate the entire row. Pull the row out and print it.

(h) Select from `data1` the names of all the respondents starting with Fred and ending with Ajax. There are a number of ways to do this.

(i) There is at least one large outlier in the sample. Add a new variable to `data1` called `log_income` that takes the natural log of the `income` variable.

3. **Lists**

(a) Take your `ideology`, `respondent`, and `income` variables and put them into a list object called `survey`. See how this structure differs from when you created `data1`.

(b) Create an object `session` that takes the value 2. Just a single number.

(c) Create an object `weeks` that is a 3×3 matrix of numbers 1 through 9. Input them `byrow`.

(d) Create a list called `dv_list` that takes `truth`, `session`, `weeks`, your dataset `data1`, and the `survey` list object. Rename these objects `truth`, `sess1`, `wk`,`dat`,and `svy` inside the `list` command.

(e) Extract the middle element from the `weeks` matrix *from* ***`dv_list`***. The command should thus start with the list itself.

(f) Change the `income` variable in `data1` *from* ***`dv_list`***. Divide `income` by 2. The command should start with the list object itself.

Tidyverse Core Functions

1. **Filter**

   (a) Load the `tidyverse` package (Actually, it's a package of packages. You'll see.) with the `library` command. If you do not already have the package installed, you can do so from the `Tools` portion of the `RStudio` menu or with the `install.packages()` command.

   (b) Load the `swiss` dataset with the `data` command. Look at the `head` of the dataset. These data represent a few key socioeconomic indicators for some of the provinces of Switzerland in the late-nineteenth century. If you want to see more about the data, use the question mark helper function `?swiss`.

   (c) Turn the dataset into a `tibble` if it is not already one. This is just practice; you do not always have to turn your data frames into tibbles.

   (d) Filter the dataset for provinces that have less that 50% male involvement in `agriculture`. Use the `pipe` operator when doing so, please. Recall: the pipe operator is a part of the `tidyverse` through the `magrittr` package.

   (e) Filter the `swiss` dataset two ways this time. Filter the data for provinces above 50% `Catholic` and for provinces and below 70% `fertility`. You should be able to do this in one "verb" command.

2. **Mutate**

   (a) Download the IMDB movie dataset onto your computer and name it `movies`. It is a large, ugly dataset with over 5,000 movies. You will have to work with datasets like this one in your life, probably often. Better to get used to it now. To import the data into **R** I recommend you try the `read_csv` function from the `readr` package. You can either code the entire path directory of the file in quotes or you can stuff a `file.choose()` function inside the `read_csv` function, *exactly* like this: `read_csv(file.choose())`.

   (b) Explore the `movie` dataset by looking at its `structure`. Then use the far superior `glimpse()` function to look at the data.

   (c) The budgets of movies are measured in individual dollars. Add a new variable to the dataset `budget2` with the `mutate` function. Make it so that `budget2` is measured in *millions of dollars*.

   (d) Who can count in minutes? Seriously. Let's add a `length` variable that is properly measured in hours, which is how normal people measure the length of movies.

   (e) Subset the `movie` dataset from "Spectre" to "Skyfall." Use the titles in `movie_title` to do this, not row and column numbers. You may need to combine some functions/commands in order to do this. This will require some function that works with characters/strings (maybe try the `stringr` package!).

3. **Plots &** `ggplot2`

   (a) Use the United Nations World Income Inequality Database (WIID). These are fresh data, updated and published in 2020. You may call the dataset whatever you like, but I suggest something like `undat` or `wiid`. Show me the `head` and `tail` of the dataset.

   (b) The 2020 version of the data has very strangely coded region and subregion levels; they are *numeric*. Recode the variable (perhaps by making new region and subregion variables) that follow the UN Geoscheme for country-region classification.

   (c) Filter the dataset for just North American countries (make sure to include Mexico). Produce a scatterplot of Gini coefficient, `gini_reported`, over time (use the `year` variable on the x-axis). Map `country` onto color. The points should look a little odd because of overplotting. Investigate the data set to find out why and tell me in a comment.

   (d) Repeat the problem above, but this time make a line plot. Does that help at all? It looks like we will have to do some summarizing in future problems to really get the most out of these data. Stay tuned.

4. **Basic statistics and summaries with** `summarize`

   (a) Use the `pipe` operator and `summarize` the Gini index `gini_reported`, the first quintile `q1` and the fifth quintile `q5`, giving me the median of each of these using one function. If you save the output as an object, make sure you also print it so that I can see it. This is a rare case where I am asking you to use `summarize` without `group_by`.

   (b) Filter the `WIID` for the year 2005 and give me the minimum values for the Gini index, the 1st quintile (`q1`), and the 5th quintile (`q5`)for the whole world.

   (c) I'm curious. For the year 2000, what are the highest values for the Gini index (`gini_reported`), 1st decile (`d1`), and 10th decile (`d10`) values reported for Africa?

   (d) Repeat the line plot from 4(d) one more time, but add one more element. Before you make the line plot, group the data by country and year and estimate a yearly mean per country using the `group_by` and `summarize` functions. This line plot should start to look more like a decent line plot.