HKS MPA-ID 2019 Pre-Math Camp Assignment

Due: 2019-08-01 on Canvas

Combined with the assigned Primers, these set of exercises get you up and running with basic data analysis in R. We realize that this is asking a lot of you, especially if you are new to programming. The Math Camp program will go over the exercise to clarify any points of confusion. That said, please feel free to contact Shiro (`kuriwaki@g.harvard.edu`) if you have any questions in the meantime.

Submission

Do your work in the rstudio.cloud environment described below and submit only the saved `.R` file to the Assignment Page in Canvas.[1] More details on how to do this are provided at the end of this assignment.

Where are we? Where are we headed?

Before you start this practice problem set, you should have completed, or at least reviewed the RStudio Primers:

- Visualization Basics
- Programming Basics
- Work with Tibbles
- Isolating Data with dplyr
- Creating Variables and dataframes

Problem 1: Familiarize with the Style Guide

Learning any language requires following its form and style. Throughout the course, we will be enforcing a set of common set of guidelines on how R code should be written. Before writing any code, read and try to internalize Book I ("Analyses") of tidyverse style guide (`https://style.tidyverse.org`), especially chapters 1 and 2.

[1] If you can't find the link, the formal link to the assignment is: `https://canvas.harvard.edu/courses/62068/assignments/285490`

Problem 2: Loading a Spreadsheet in RStudio

In the primers, you interacted with R via a web interface. That made each task clear and immediate, but it also restricted you from using all of R's features. In fact, in most of your work you will be working with R on a different interface called RStudio. First, let's get you set up on the interface. We will go into more depth in class.

1. **Create a rstudio.cloud account**: On the internet, go to rstudio.cloud. You will be prompted to Sign in or make an account. Please create one. You will use this account for prefresher and you may choose to use it for your classes. Therefore, we advise you use your HKS account.

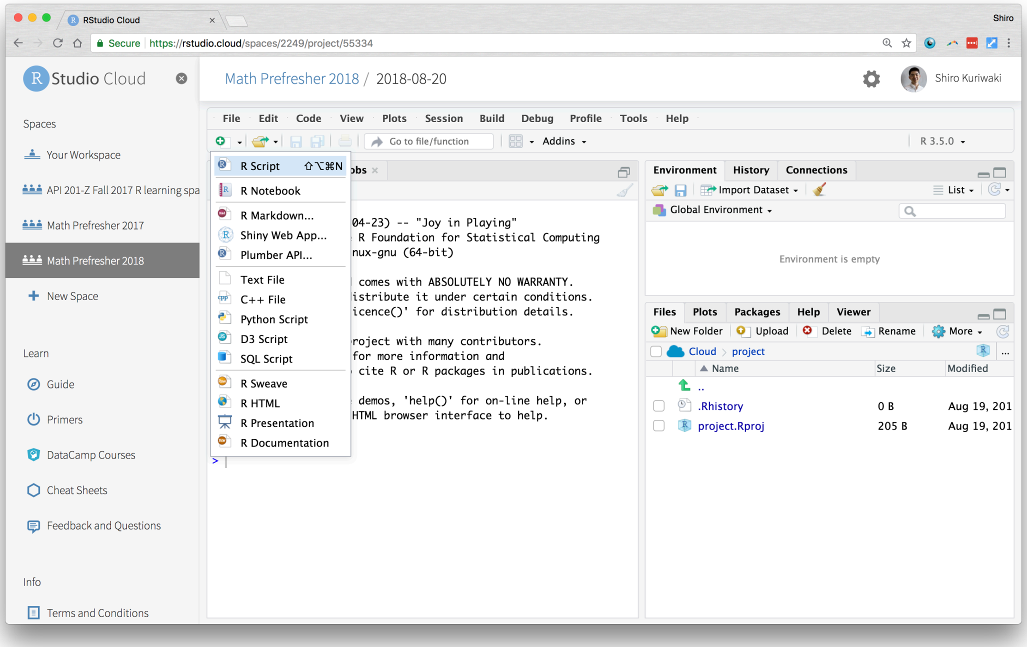
2. **Sign into the Class Space**: Once you have signed in, join the space we have created for Math Camp. Use this access link `https://rstudio.cloud/spaces/18236/join?access_code=pR6TvDKi39LKuDHl%2BfltWf2nGC%2Fb0VAk4TZ1Kz5i` and make sure your account is listed as a member.

3. **Copy a Project** In the projects tab, go to the Assignment `01_Summer-Assignment`, and click "Start".

4. **Understanding the GUI and R the program**. It will take 30 seconds to about a full minute for a new window to finish loading. Welcome to RStudio!

   RStudio is a type of **GUI** (Graphical User Interface) for R, which is a programming language. A GUI allows users to interface with the software (in this case R) using graphical aids like buttons and tabs. To most computer users, everything is a GUI (like Microsoft Word or your "Control Panel"). RStudio is also an "IDE" (Integrated Development Environment) meaning that it provides shortcuts to advanced tools for working with R.

   The **Console** is kind of a the core window through which you see your GUI actually operating through R. It's not graphical so might not be as intuitive. But all your results, commands, errors, warnings.. you see them in here. A console tells you what's going on now.



```
04-23) -- "Joy in Playing"
 R Foundation for Statistical Computing
nux-gnu (64-bit)

 comes with ABSOLUTELY NO WARRANTY.
istribute it under certain conditions.
icence()' for distribution details.

roject with many contributors.
for more information and
 cite R or R packages in publications.

 demos, 'help()' for on-line help, or
HTML browser interface to help.
> 
```

▲ Name	Size	Modified
..		
.Rhistory	0 B	Aug 19, 201
project.Rproj	205 B	Aug 19, 201

5. **Open a Script**: From the Toolbar's `File`, click to `New File`, then `R Script`. This will create a blank text file with the `.R` file extension. Please enter and edit all your code for this assignment in this file, and submit the saved version (more on the requirements at the end). In programming, we call this type of file a "script". It is a plain (i.e. no formatting added on) text file with code that is immediately executable.

6. **Read in a Dataset**: Now, let's import an external dataset. As a data analyst, you will almost always obtain and clean up your own dataset.

   Let's read in a dataset which we'll call the World Economic Outlook dataset. Here, you'll first rely on the convenience features that RStudio provides.

   At the bottom right corner, you should see a "Files" tab. Click through to the folders `data`, then `input`, and click on the filename `WEO-2018.xlsx,"` Choose Import dataset (Figure 1). This starts the process of structuring a piece of R code to read a flat file. One thing you want to change is to make the name of the imported dataset informative, as recommended in the style guide (Figure 2).

   You should see a preview of the spreadsheet and the command that produces it (Figure 3). The bottom-right button, "Import", will send the code directly into the Console.

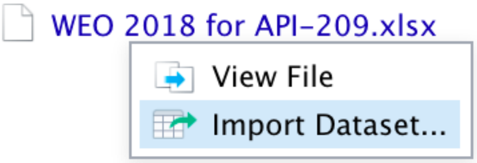
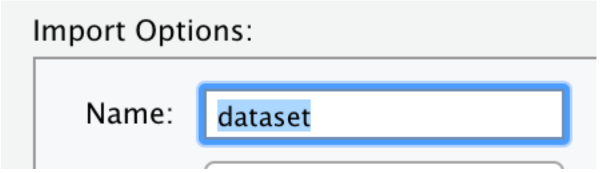
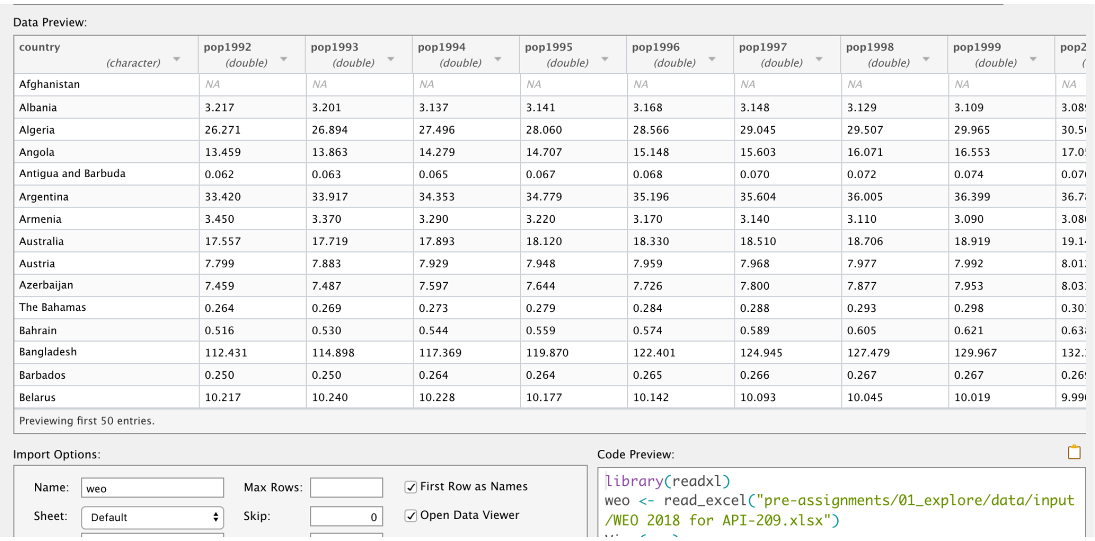


Figure 1: Importing a file by click-and-drag


Import Options:

Name: dataset

Figure 2: Assign an informative name to your dataset


Data Preview:

country	pop1992	pop1993	pop1994	pop1995	pop1996	pop1997	pop1998	pop1999	pop2
(character)	(double)	(double)	(double)	(double)	(double)	(double)	(double)	(double)	(
Afghanistan	NA	NA	NA	NA	NA	NA	NA	NA	NA
Albania	3.217	3.201	3.137	3.141	3.168	3.148	3.129	3.109	3.08
Algeria	26.271	26.894	27.496	28.060	28.566	29.045	29.507	29.965	30.5
Angola	13.459	13.863	14.279	14.707	15.148	15.603	16.071	16.553	17.0
Antigua and Barbuda	0.062	0.063	0.065	0.067	0.068	0.070	0.072	0.074	0.07
Argentina	33.420	33.917	34.353	34.779	35.196	35.604	36.005	36.399	36.7
Armenia	3.450	3.370	3.290	3.220	3.170	3.140	3.110	3.090	3.08
Australia	17.557	17.719	17.893	18.120	18.330	18.510	18.706	18.919	19.1
Austria	7.799	7.883	7.929	7.948	7.959	7.968	7.977	7.992	8.01
Azerbaijan	7.459	7.487	7.597	7.644	7.726	7.800	7.877	7.953	8.03
The Bahamas	0.264	0.269	0.273	0.279	0.284	0.288	0.293	0.298	0.30
Bahrain	0.516	0.530	0.544	0.559	0.574	0.589	0.605	0.621	0.63
Bangladesh	112.431	114.898	117.369	119.870	122.401	124.945	127.479	129.967	132.
Barbados	0.250	0.250	0.264	0.264	0.265	0.266	0.267	0.267	0.26
Belarus	10.217	10.240	10.228	10.177	10.142	10.093	10.045	10.019	9.99

Previewing first 50 entries.

Import Options:

Name: weo
Max Rows:
☑ First Row as Names
Sheet: Default
Skip: 0
☑ Open Data Viewer

Code Preview:

```
library(readxl)
weo <- read_excel("pre-assignments/01_explore/data/input
/WEO 2018 for API-209.xlsx")
```

Figure 3: Import helper

Problem 3: Finding top and bottom

The following questions are based on the latest version of the World Economic Outlook database published by the International Monetary Fund (IMF), which you just read in.

First, familiarize yourself with the spreadsheet, which contains total GDP adjusted for purchasing power parity and total population for each country in the database. Then import these data into R and perform the following analyses. Note that GDP values are in millions of 2011 international dollars, so you can directly compare values in different years. Population data is in millions of persons.

(1) Write a command (connected by pipes) that (i) first sorts the dataset from lowest to highest real GDP in 2017, and then (ii) outputs a two-column dataset of the country and its GDP.

(2) Write a command that is the same as (1) but now sorts it in descending order of 2017 GDP (highest to lowest).

(3) Write a command that shows African countries in descending order of their 2017 GDP (Use the variable `continent` to filter on African countries).

Problem 4: GDP per capita

Create a new tibble object called `weo_percep` that includes:

- A variable called `gdp_percap_2017` that is the country's GDP per capita in 2017,
- A variable called `gdp_percap_1992`, which is the same as above but for 1992, and
- A variable caled `growth_2017_1992` which indicates the difference between the two variables above, with positive values indicating growth.

Problem 5: Graphing

Make a scatterplot that shows a countries 1992 GDP per capita on the x-axis and its 2017 GDP per capita on the y-axis.

You might notice that the scatterplot itself is not as informative as it could be. In math camp, we will spend a session discussing the nuts and bolts of making a high-quality graphic that is informative and user-friendly.

Problem 6: Mean and Median

(1) Write code that reports the mean of country-level GDP per capita of 1992 in one column and the mean for 2017 in another. Make sure that column names are self-explanatory.

(2) Do the same, but showing the median instead of the mean. Make sure to ignore any missing values in the calculation of the median, so a value is returned.

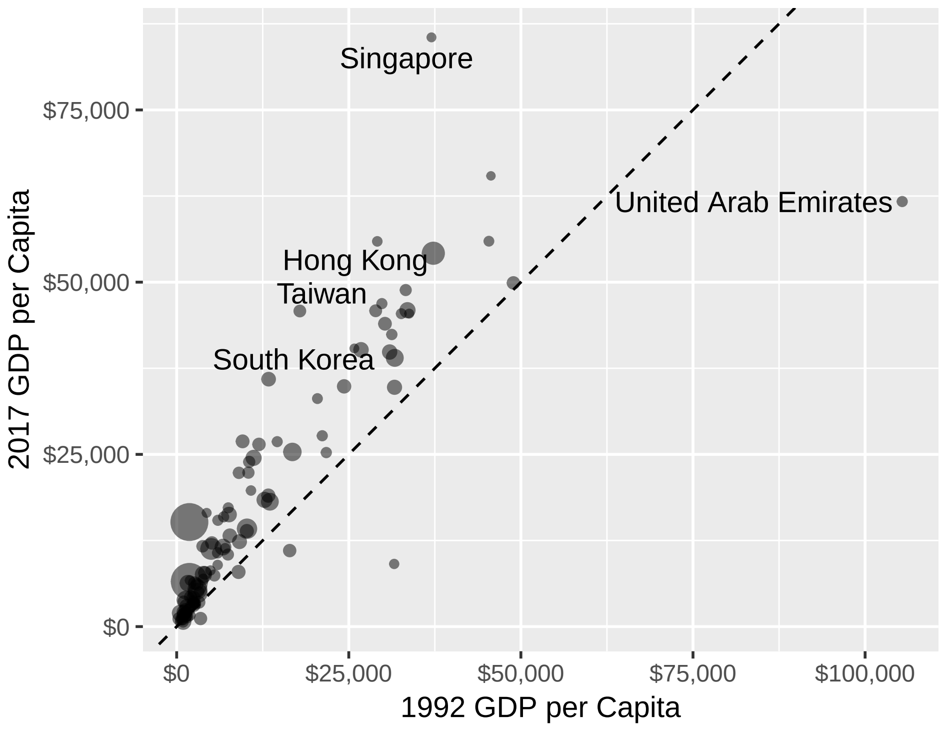
Problem 7: slice() and filter()

Much of data analysis is understanding how new tools and functions work. Here, we introduce a new function that may come in handy — the function `slice()` is part of the tidyverse and allows you to filter rows by their position. Notice that `slice()` and `filter()` are similar in that they subset rows of a dataset, but differ in the types of input they require — the former asks for positions, the latter asks for conditions.

Check out the help page of `slice`. Then, write a command that shows the countries with the top three and bottom three 2017 GDPs, thereby combining the output in the first two R exercises.

Optional Challenge Problem

Make a graph like the one shown in Figure 4. Follow both the graphical components of the graph shown, as you see them, as well as the description of the measures as described in the caption.



Points sized by 2017 population.
Labels show top 5 countries with the most distance from the 45−degree line.
Only countries with population at least 5 million in 1992 or 2017 shown.

Figure 4: Changes in GDP per capita between 1992 and 2017.

Submitting

Once you have completed or made an attempt for all the problem, please clean up your R script, download it from the cloud, and submit it to Canvas.

Math camp insturctors will check and provide comments for your code. You should follow these guidelines to clean up your final submission (and should do so for all future scripts):

- Delete any failed attempts or duplicative code.
- Label the relevant question number by comment (e.g., `## Problem 1.1`. Follow the style guide for the exact format)
- Try restarting (Toolbar `Session` > `Restart`) and running your entire code at once (e.g., Select All Text and Run, or `Run All` by `option` + `command` + `R`. This ensures that your code is replicable.
- Follow other guidelines from the style guide, such as putting the `library()` command at the beginning of the script.

To help us sort through all submissions, please name your script with your last name. e.g., `kuriwaki_R-assignment.R`.

After editing your code, save it to the main project folder, and then download it by right-clicking the file icon (in the File Pane), and selecting `Export` (Figure 5). Download the script and attach it to your Canvas submission.

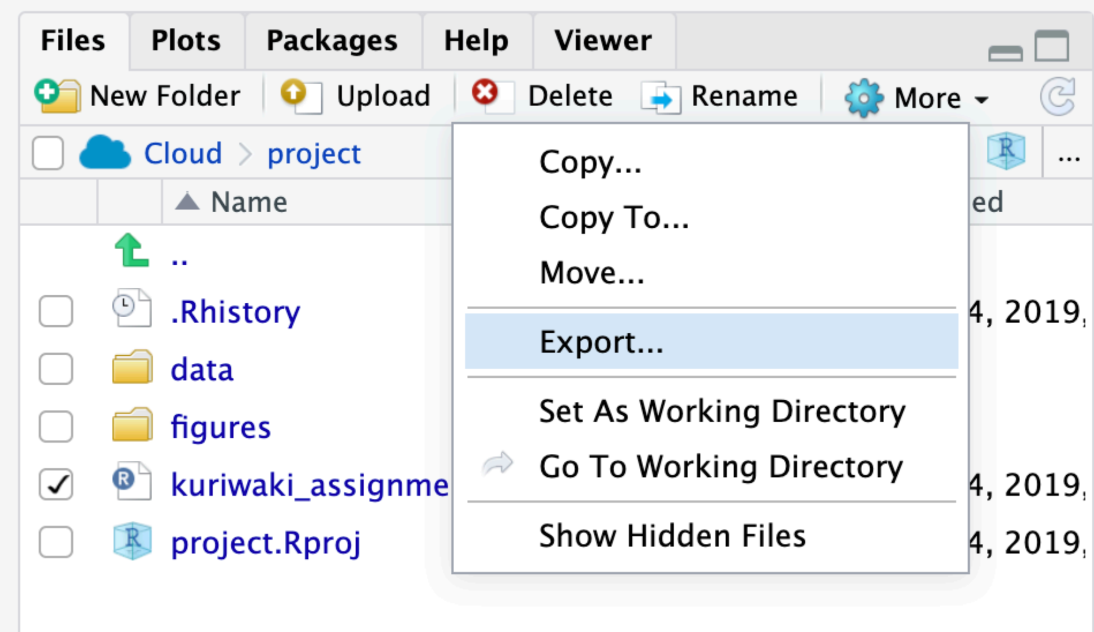


Figure 5: Downloading your final code