R Summer Assignment for API-209

Summer 2019, MPA/ID Incoming Class

Due Augst 1, 2019

Combined with the assigned primers, this set of exercises will help you get you up and running with doing basic data analysis in R. The Math Camp program will go over the exercise to clarify any points of confusion.[1]

Submission

Do your work in the rstudio.cloud environment described below and submit only the saved `.R` file to the Assignment Page in Canvas.[2] More details on how to do this are provided at the beginning and 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

After this assignment, we'll be offering several more sessions in Math Camp to cover more tools and concepts.

[1] That said, please feel free to contact Shiro (kuriwaki@g.harvard.edu) if you have any questions in the meantime.

[2] Once you are authorized to access the MPA/ID Math Camp 2019 page, the direct link to the assignment is: `https://canvas.harvard.edu/courses/62068/assignments/285490`

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.

Problem 2: Loading a Spreadsheet in RStudio

The interactive windows in the primers got you started in R, but was also restrictive. Most of your data analysis work will involve programming in R on a designated interface called RStudio. Follow the steps below to get set up and load a dataset.

1. **Create a rstudio.cloud account**: On the internet, go to `https://rstudio.cloud`. Please create a new account for yourself. You will use this account for math camp, so we advise you use your HKS email.

2. **Sign into the Class Space**: Once you have signed in, join the Math Camp "space" through the access link `https://rstudio.cloud/spaces/18236/join?access_code=pR6TvDKi39LKuDHl%2BfltWf2nGC%2FbOVAk4TZ1Kz5i`. By joining this group, you can access R material shared with the group.

3. **Copy a Project** In the projects tab, go to the Assignment `01_Summer-Assignment` (Figure 1(a)), 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 (Figure 1(b)). Welcome to RStudio!

   RStudio is a *GUI* (Graphical User Interface) for the programming language R. A GUI allows users to interface with the software using graphical aids like buttons and tabs. Most daily software is a GUI (like Microsoft Word or the 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 the core window through which you can observe R operating (through the GUI). All your results, commands, errors, warnings get shown here.

5. **Open a Script**: From the Toolbar's `File`, click to `New File`, then `R Script` (Figure 1(c)). This will create a blank file with the `.R` file extension. Please enter your code for this assignment in this file, and submit it (see the end of this assignment for more details). We call this type of file a "script". It is a plain (i.e. no formatting added on) text file with executable code.

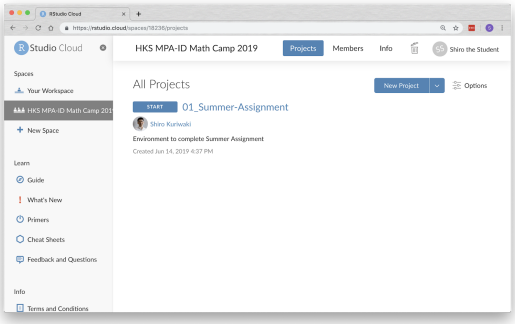
6. **Read in a Dataset**: Here, we'll first rely on the convenience features that the GUI provides. At the bottom right corner, you should see a "Files" tab. Click through to the folders `data`, then `input`, and click on the filename `WEO-2018.xlsx`." Choose `Import Dataset...` (Figure 1(d)). This starts the process of structuring a piece of `R` code to read in the file. One thing you want to **change** is the name you assign

to your imported dataset. Because you will be typing in the object name many times. Pick a short and informative name (like `weo`), as recommended in the style guide (Figure 1(e)).

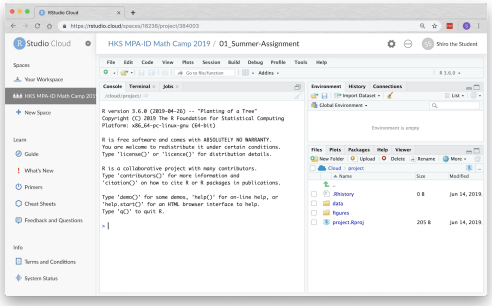
You'll see a preview of the spreadsheet and the command that produces it (Figure 1(f)). The bottom-right button, "Import", will send the code directly into the Console. To make your script replicable, copy the first two lines of this inserted code (the `library(readxl)` command and the line that involves `read_excel()`) to the beginning of your script.

All objects created in R will appear in the "Environment" tab (top-right), along with information like variable names. After you have imported the dataset, it can also be available for browsing on a tab right next to your R script.

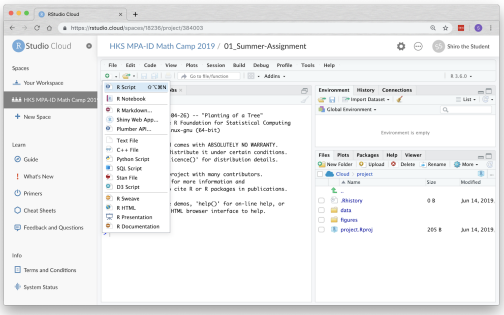
(a) Open the Assignment Project



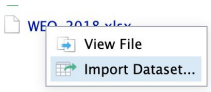
(b) Opened Assignment in the RStudio GUI/IDE



(c) Open a New R Script



(d) Navigate to the data file and Import



(e) Change the assigned name to an informative one

(f) Preview the dataset before completing the Import

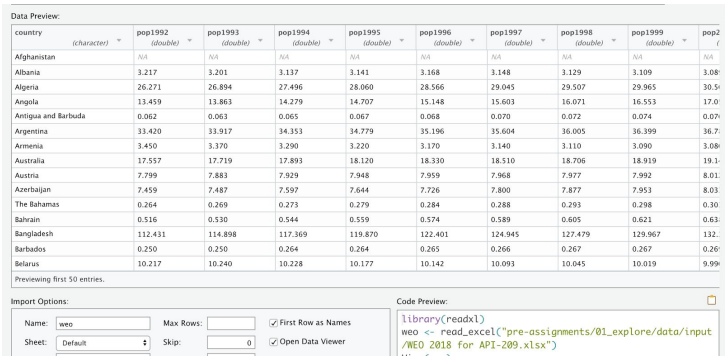


Figure 1: Example Screenshots Corresponding to Problem 2

Problem 3: Sorting by Values

The following questions are based on the latest version of the World Economic Outlook dataset published by the International Monetary Fund (IMF), which you just read in. Each row in the spreadsheet is a country, with total GDP for a given year adjusted for purchasing power parity (with the `rdgdp` column prefix) and total population (with the `pop` column prefix). GDP values are in millions of 2011 international dollars, so you can directly compare values in different years. Population values are 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. To see if your code works and see its output, highlight your code in the R script and click on the "Run" Icon (or the hotkey `command` + `Enter`).

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

(3) The `arrange()` command can sort on more than one variable. To rank countries within their continent, write a command that sorts the countries by continent (in alphabetical order), then by GDP per capita in descending order. Remember different inputs ("arguments") to a function are separated by commas.

(4) 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 is the same as `weo` but also 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 called `growth_2017_1992` which indicates the **rate** of growth between the two variables above, following the definition below.

Operationalize the rate of growth between two periods a and $b > a$ as:

$$\text{growth}_{b,a} = \frac{\text{GDP per captia}_b - \text{GDP per captia}_a}{\text{GDP per captia}_a}$$

Problem 5: Graphing

(1) 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.[3]

[3] 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

(2) Show the same figure, but coloring the points by continent. That is, countries of the same continent should have the same color.

(3) Show the same figure, but assigning a different shape of point for different continents. Also, set the color to all offices to navy by using the color label `"navy"`.

Problem 6: Mean and Median

(1) Let's start to think about summarizing variables using summary statistics. Write code that reports the mean of country-level GDP per capita of 2017 in one column and the median for 2017 in another. Pick column names that are sufficiently self-explanatory but also concise.

(2) Write code that indicates which countries have missing values for 1992 GDP per captia and 2017 GDP per capita.

(3) Try the same as (1) but now with replacing 2017 with 1992. You should initially notice that you get a missing value for both summary statistics. This is because by default, `mean()` and `median()` report a missing value if at least one of its input values is missing, and as you probably found in part (2), some countries have have missing values for 1992 GDP per capita. Now, modify your code so that you change this default and ignore the missing values in your computation. As the help page for the functions indicate, the relevant argument is `na.rm` (for NA - remove). Change its logical value from `FALSE` to `TRUE`, while making sure to follow the style guide for proper spacing (style guide section 2.2.3).

Problem 7: slice() and filter()

Much of data analysis is understanding how new functions work through reading the documentation and experimentation. 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` (recall, e.g., by typing `?slice` in the Console). 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 and Challengin] More Graphing

Note: This problem is optional, it involves some commands not covered in the primers.

user-friendly.

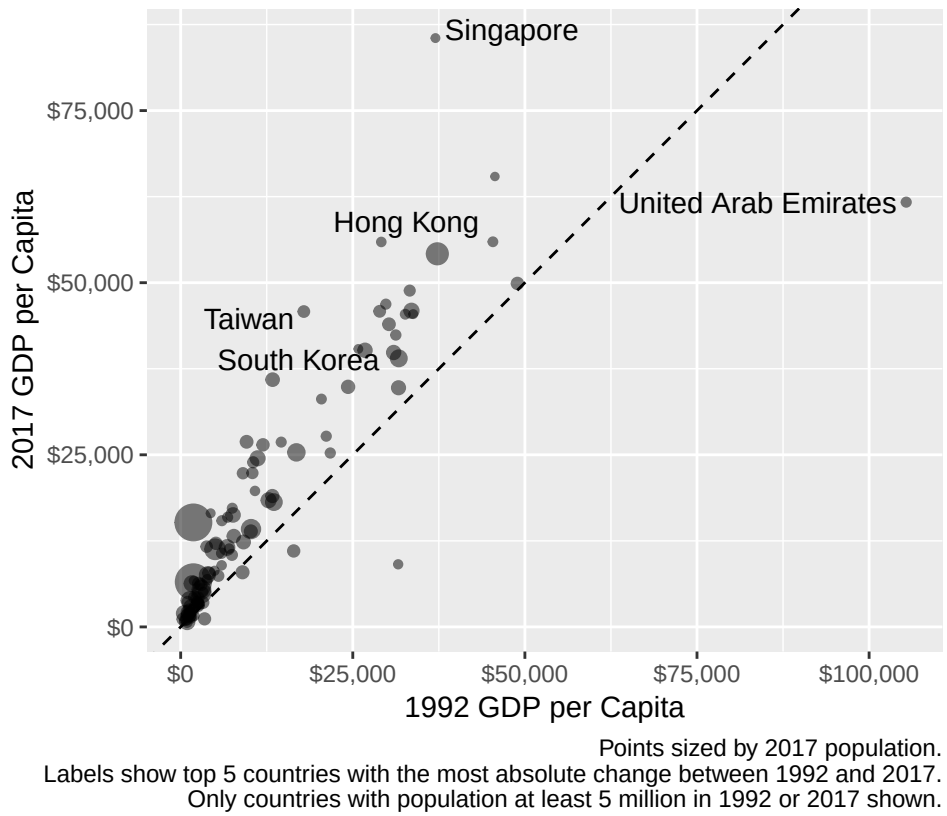


Figure 2: Changes in GDP per capita between 1992 and 2017.

Make a graph like the one shown in Figure 2. 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 Figure caption. *Hint:* Check out the packages `ggrepel` and `scales` to implement some of the features.

Submitting (and Survey)

Before you submit, please complete this brief survey as so that we understand better you background and can design the R activities in math camp accordingly.

Placeholder https://harvard.az1.qualtrics.com/jfe/form/SV_dhY1CK6Y58PSvqt

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 instructors 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`. Before you do this, though, make sure you explicitly load the `tidyverse` package in your code by adding `library(tidyverse)` to the beginning of your file. This ensures that your code is replicable.
- Follow other guidelines from the style guide, such as breaking up long lines and properly using spaces.
- To help us sort through all submissions, please name your script with your last name followed by your first name. e.g., `kuriwaki_shiro.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 3). Download the script and attach it to your Canvas submission.

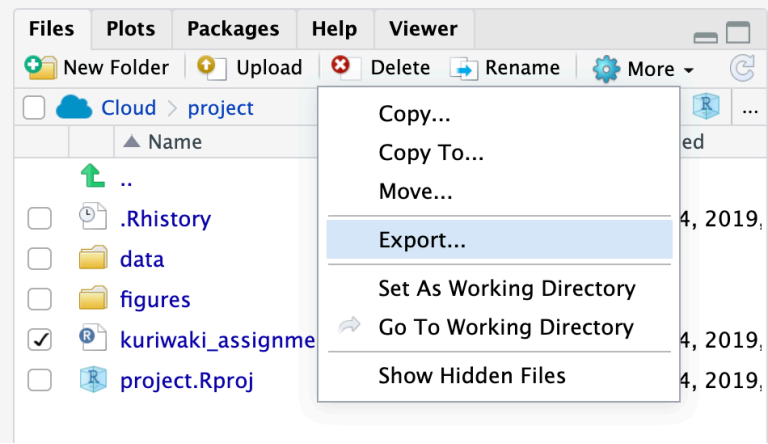


Figure 3: Downloading your final script