EECS 598 and EECS 498 (Fall 2021) – Assignment 0
Due date: Monday, September 6 at midnight EST

Collaboration. You are free to discuss the assignment with others or work together towards the solution. However, all of your code must be written by yourself. Your write-up must be your own as well. **Please list your collaborators at the top of your submissions.**

Goal. The goal of this assignment is to help you get familiar with necessary tools and libraries that subsequent assignments will build upon. In particular, in this assignment, you will have access to a code base that you will use to build upon for future assignments.

1 Code base

Download the code base from
`https://web.eecs.umich.edu/~xwangsd/courses/f21/assignments/codebase.zip`.

This is a Visual Studio Code project written in Java. Therefore, the easiest way to build the project is to use Visual Studio Code. Make sure you have Java (1.8 or higher) installed. The project comes with a build script `pom.xml`. That means, you can also build this project from your terminal using `maven`.

This is the structure of the `codebase` folder.

```
codebase
├── src
│   └── main
│       └── java
│           └── syn
│               └── base
│                   ├── AST.java
│                   ├── ASTNode.java
│                   ├── CFG.java
│                   ├── Dataframe.java
│                   ├── Interpreter.java
│                   ├── Production.java
│                   └── Synthesizer.java
├── lib
│   ├── com.microsoft.z3.jar
│   ├── libz3.dylib
│   └── libz3java.dylib
└── pom.xml
```

For this assignment, the most important classes are `Dataframe.java` and `Interpreter.java`.

- `Dataframe.java`: since we're going to deal with R programs that transform tables/dataframes, this is a key data structure that models a table/dataframe.

- `Interpreter.java`: this class provides key functions for evaluating R programs. In particular, in this assignment you will use `eval(String, Dataframe)`. This function takes as input a `String` that encodes an R program and a `Dataframe` that represents the input table. It returns a new `Dataframe` that corresponds to the program's output. We will only consider R programs that take as input *one single* table, however, you can definitely extend the code base to support multiple input variables.

Below is a very brief explanation of other classes which we will explain in more detail in future assignments when they become more important.

- `AST.java`: we will represent R programs as abstract syntax trees (ASTs).
- `CFG.java`: this is a context-free grammar (CFG) that is used to define the search space.
- `Synthesizer.java`: this is an abstract synthesizer that you will work on in future assignments.

There is also a `lib` folder. We will use these libraries in future assignments, but not this one.

2 Getting started

To get started with this assignment, download `pa0` from
https://web.eecs.umich.edu/~xwangsd/courses/f21/assignments/pa0.zip.

You should be able to find a folder called `pa0`. Put `pa0` in the `syn` folder.

```
codebase
└─ src
   └─ main
      └─ java
         └─ syn
            ├─ base
            └─ pa0
               ├─ Test0.java
               └─ Test1.java
```

Run the `main` function in `Test0.java`. It should crash at some point. This is expected, because we haven't yet set up the environment. We will do that shortly.

3 R basics

R is a programming language [2] that is widely used among statisticians and data scientists for developing statistical software and data analysis. In this assignment (and subsequent assignments), we will mainly focus on two packages, namely `tidyr` [3] and `dplyr` [1], which are commonly used for data manipulation.

To get started, download and install R: https://cloud.r-project.org/. Once installed, start the R Console application. Install the `tidyr` from R Console by typing `install.packages("tidyr")`. Do the same for `dplyr`, too. Then, import these packages, e.g., by typing `library(tidyr)`.

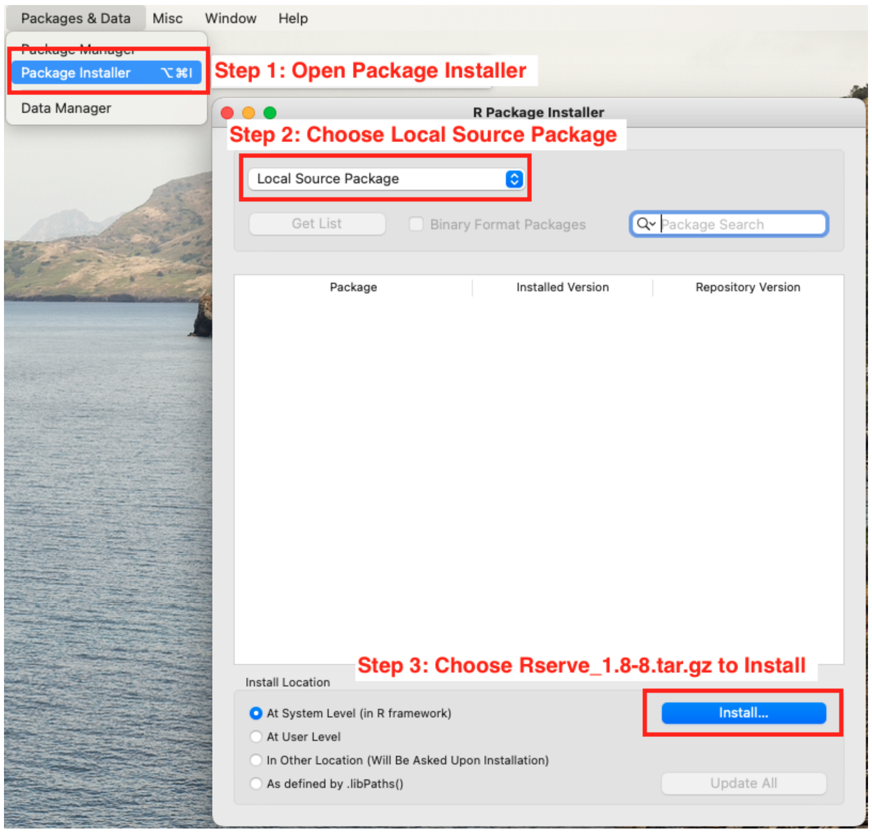
Now, you should be able to run R programs that use functions from `tidyr` and `dplyr` packages from R Console. For instance, try some examples in this blog post:
https://blog.rstudio.com/2014/07/22/introducing-tidyr/

4 Rserve and Rengine

R Console allows you to run R programs *from the console*, but we eventually want to run R programs from *Java*. We will do this using `Rserve` and `Rengine`. `Rserve` is a TCP/IP server that allows one to integrate the R back-end in other applications—in our case, a program synthesizer—without the need to initialize R or link against R libraries. Since our program synthesizer will be written in Java, we use `Rengine` to support the communication between our Java programs and `Rserve`.

Follow these steps to set up the environment.

- Download `Rserve_1.8-8.tar.gz` from https://www.rforge.net/Rserve/files/.
- Install this package from R Console as shown below.



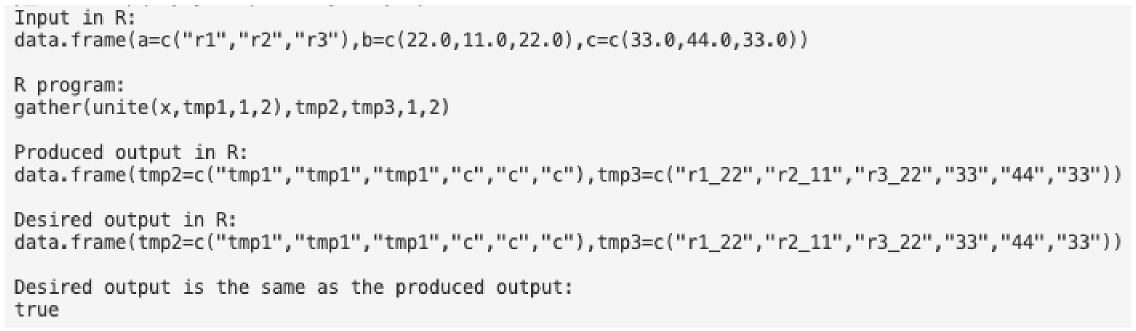
- Import the `Rserve` package from R Console: `library(Rserve)`
- Start the server from R Console: `Rserve(args="--no-save")`

Now, you should be able to see a message in R Console "Rserve started in daemon mode". That means your server is up running. If you close the R Console, the Rserve process may still be active. You will need to kill the Rserve process manually.

Note that we used Rserve version 1.8-8 and that is only because it matches the `Rengine` version used in `pom.xml`. It is possible a different version also works. Furthermore, the aforementioned process was tested on a MacOS machine. If you use a different OS, consider using a virtual machine running MacOS or you may need to figure out how to set up the environment on your specific machine. For example, on Linux there is no GUI installer, but you can type the following in R Console to install Rserve.

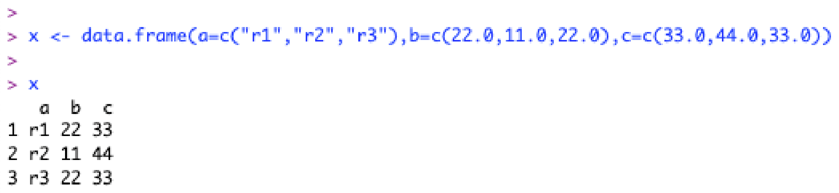
```
packageurl <- "https://www.rforge.net/Rserve/snapshot/Rserve_1.8-8.tar.gz"
install.packages(packageurl, repos=NULL, type="source")
```

Now, let's move on to the `Rengine` part. If you use the provided `pom.xml` script to build the project, you should be able to directly run the `main` function in the `Test0` class without any errors. However, if you chose to build the project yourself, you may need to download the `Rengine` library and make sure it's added into your `CLASSPATH`. Once you set up all these, running `main` in `Test0` should no longer crash and it should print the following message:



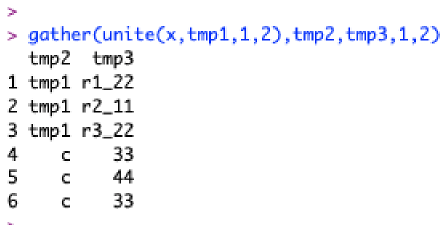
```
Input in R:
data.frame(a=c("r1","r2","r3"),b=c(22.0,11.0,22.0),c=c(33.0,44.0,33.0))

R program:
gather(unite(x,tmp1,1,2),tmp2,tmp3,1,2)

Produced output in R:
data.frame(tmp2=c("tmp1","tmp1","tmp1","c","c","c"),tmp3=c("r1_22","r2_11","r3_22","33","44","33"))

Desired output in R:
data.frame(tmp2=c("tmp1","tmp1","tmp1","c","c","c"),tmp3=c("r1_22","r2_11","r3_22","33","44","33"))

Desired output is the same as the produced output:
true
```

You can copy the input to the R Console and assign it to a variable x as shown below.



```
>
> x <- data.frame(a=c("r1","r2","r3"),b=c(22.0,11.0,22.0),c=c(33.0,44.0,33.0))
>
> x
   a  b  c
1 r1 22 33
2 r2 11 44
3 r3 22 33
```

Then, run the R program from R Console.



```
>
> gather(unite(x,tmp1,1,2),tmp2,tmp3,1,2)
  tmp2  tmp3
1 tmp1 r1_22
2 tmp1 r2_11
3 tmp1 r3_22
4    c    33
5    c    44
6    c    33
```

As you can see, the output from R Console is the same as the output from the Java project. As a sanity check, `Test0` also says the produced output is indeed equal to a manually constructed `Dataframe` object.

5 Your work

In this assignment, you will write your own test case in `Test1.java` in the `pa0` folder. Simply follow what's in `Test0.java`, however, choose your own input dataframe and write your own R program.

6 Submission

Your Task. There is one task for you to complete. That is, write your own test case in `Test1.java`, run it, and take a screenshot of the output.

Canvas. Submit the following in one single .zip file:

- (50 points) **Your code** including **only** the `Test1.java` file.
- (50 points) **Your report** in PDF that contains your screenshot. You may optionally include your explanation as well.

References

[1] The dplyr package. https://dplyr.tidyverse.org/.

[2] The R Project for Statistical Computing. https://www.r-project.org/.

[3] The tidyr package. https://tidyr.tidyverse.org/.