CADS Assignment 2 - Creating and Manipulating a Data frame in R

Instructions:

1. With your partner, please work through this Assignment.
2. First part (Guided Practice) has a lot of help.
3. Guided Practice #1-5 carry 10 points each.
4. You should be able to complete Exercise 1 and 2 with some help
5. Exercise #1 and 2 carry 25 points each.
6. Use the web resources that I provided or look for your own.

Suppose you generate data in your own research, how are you going to use R to analyze your data? You have two options: You can input your data to excel and then import it into R or you can input your data directly into R. In this assignment, we will practice step-by-step how to use the latter option. We will use simple examples with just a few rows and columns. That is all that is needed. Once you know what to do with a small dataset, you are empowered when you do your own research. The first option will be left for Assignment # 3.

PS: Most of the time, data will be entered into a spreadsheet like Excel first. Adding rows, inserting columns are very easy. For analysis and visualization, R is then used to take advantage of portability and reproducibility which is now required by many publications.

Again, there are several ways to accomplish a task. We will usually show one way. If you google, you will find others. We are limited if all we have is Base R. But usually, when we add libraries, we have more options. Here, we will add a library called “dplyr”. Some people prefer to add “tidyverse” as it

library(dplyr) # Needed for some functions to work.

Guided Practice 1

first_column<- c(“value_1”, “value_2”,…) # Create what goes into column 1.

second_column<- c(“value_1”, “value_2”,…) #Create what goes into column 2.

df <- data.frame(first_column, second_column) # Combine the two columns to form a data frame.

Use the table to create a data frame in R

Name	Age
Jon	23
Bill	41
Maria	32
Ben	68
Tina	26

Let us use the Template to create the data frame as follows and call it df::

```
Name <- c(“Jon”, “Bill”, “Maria”, “Ben”, “Tina”)  # Notice that we put the name in quotes because these are characters and we want them exactly as written.

Age <- c(23, 41, 32, 68, 26)

df <-data.frame(Name, Age)

print(df) # Remember that you want to avoid print if dealing with a large data frame. You use head() or tail() which shows 6 rows.
```

When you run the code (select the entire code and run) in R, you should get the DataFrame:

```
   Name   Age

1  Jon    23
```

```
2   Bill    41

3   Maria   32

4   Ben     68

5   Tina    26
```

Notice that the numbers 1,2,3…are automatically generated as in Excel.

Now, use your knowledge to extend the data frame

Guided Practice 2: Extend the work

a) Change the column with Name to First_Name,

b) Add a column and call it Last_Name, so that we now have Jon Smith, Bill Kemp, Maria Doe, Ben William, Tina Walker.

c) Add a column and call it Gender, using Jon as M, Bill as M, Maria as F, Ben as M and Tina as F.

d) We can do b) and c) together if we want or do it one at a time.

e) Give a new name to the modified DataFrame, let's call it Myfinal_df

f) Take a look at Myfinal_df by using view(), head(), print() # Remember that if it is a large data frame, you want to avoid printing.

Observation: It is necessary to place quotes around text (for values under the name column) but not required for numeric values placed under the Age column.

We can add new rows to our data frame. If you are doing research, you may interview more people and so have more observations. There are several ways to do this but we will describe just one. We keep it simple by adding one row at a time.

There are always several ways to get to our goal and you just google to find the right code. Whatever you are trying to do, someone has done something similar. You can use their code and modify accordingly and move on with your own analysis.

<u>Guided Practice 3</u>

a) Add a row to the above data frame for Michael, and age 34.

df_plusOnerow <- rbind(df, c(“Michael”, 34)) # Notice I went back to df which we did not overwrite. We could use My_df (another exercise if you wanna do that). Notice the suggestive name-a guy created a function called rbind (r for row obviously) which is nice.

df_plusOnerow # This prints our updated data frame.

b) Add several rows: Michael, 34 and Jane 42. Here we just add two rows.

Obs: Let's go back and use the original data frame df (df still retains old input as we did not reassign).

New_persons <- data.frame(Name=c(“Michael”,”Jane”), Age=c(34,42))

df_plusNewrows <- rbind(df, New_persons)

# Notice that rbind simply takes two tables and combines them since the two have the same variables.

df_plusNewrows # This prints out our updated data frame with 2 new rows.

Observation: As you work, you always have the option to overwrite your data or create new names and assign your updated data frame to the new name. The latter option is always preferred in case you need to start over. In any case, you should always preserve your original data. NEVER overwrite your original data.

We can append new columns (this will happen when you want to add new variables, or you compute new variables based on current variables). We will describe two ways to do this.

Notice: We go back to the original data frame df (just to make things clearer. You may want to practice building on the last data frame which now has 8 observations since we added two.

Guided Practice 4:

a) Add last name column to df, add Gender column to df. Call the new data frame My_ Newdf. We do this in two steps:

New_df <- data.frame (Gender= c(“M”, “F”, “M”, “F”, “M”), Last_Name= (“Smith”, “Kemp”, “Doe”, +“William”, “Walker”)

# the ‘+’ means we didn't complete our “sentence”; obvious since our parentheses are not balanced.

My_Newdf <- cbind(df, New_df)

b) Change the variable Name to First_Name, use the rename() function.

# You should make sure to install and load dplyr as follows: install.packages(“dplyr”)

# Make functions in dplyr accessible by executing: library(dplyr)

# By doing those two things, we make sure that the rename() functions are available for us to use.

My_Finaldf <- rename(My_Newdf,First_Name=”Name”)

My_Finaldf # Check out your final product or print it.

c) Interchange the Gender and Last_Name columns

Guided Practice 5

We can make changes to rows as well: Here a few common tasks and sample template:

df <- data.frame(x1=11:14, x2=2:5, x3=9) # create example data

df # Print example data

```
     x1  x2  x3

  1  11  2   9
```

```
    2   12   3     9

    3   13   4     9

    4   14   5     9
```

a) Remove a row:

df1 <- df[2,] # Extract one row of data by number (second row in this case). The column number is empty.

df1   # print df

```
         x1    x2    X3

    2   12   3     9
```

b) Remove multiple rows:

df2 <- df[c(1,4),]   # Extract multiple rows

df2   #print df

```
         x1    x2    x3
```

```
  1  11  2   9

  4  14  5   9
```

Guided Practice 6:

Df <- data.frame(x1=c(3,7,1,8,5), x2=letters[1:5], group=c(“g1”, “g2”, “g1”, “g3”, “g1”) # create an example. Note that we use a different data frame, because Df is different from df.

Df   # Print data frame

```
     x1  x2   group

 1   3   a    g1

 2   7   b    g2

 3   1   c    g1

 4   8   d    g3

 5   5   e    g1
```

a) Subset rows with == (logical equal)

Df1 <- Df[Df$group==”g1”,]   # Subset rows

```
Df1   # print df

      x1   x2   group

 1    3    a    g1

 3    1    c    g1

 5    5    e    g1
```

b) Subset with != (logical not equal to)

```
Df2 <- Df[Df$group  != “g1”, ]    # Subset with not equal !=

      x1   x2   group

 2    7    b    g2

 4    8    d    g3
```

c) We can use the subset() function as follows:

```
Df3 <- subset(Df, group==”g1”) # We can include multiple conditions. Google to find templates.

# We must load dplyr() to be able to use the subset() function..
```

	x1	x2	group
1	3	a	g1
3	1	c	g1
5	5	e	g1

d) Subset using multiple conditions (Try this)

subset(Df, group==”g1” & x2>3) # This asks to extract rows with column g1 and x2 bigger than 3.

# Check out other functions-select(), filter(), transform(), mutate()

# Remember to load dplyr using the library(dplyr).

<u>Exercise 1</u>

Use the tools given above and google others for the following exercises:

Create the following data frame, my_df:

43	181	M
34	172	F
22	189	M

27	169	M
35	161	F
48	158	F

a) With Row names: John, Jessica, Steve, Rachel, Alice, Debby.

b) Add Column names; Age, Height, Gender. Retain the data frame as my_df.

c) Find the structure of the data frame-observations, variables, class. Hint: Use str(my_df)

d) Summarize the data. Hint summary(my_df)

e) The average age and average height in my_df separately.

f) The average age for Male and average age for Female.

g) Change the row names of my_df so that the data becomes anonymous: Use Patient1, Patient2, Patient3 instead of actual names. Then call the modified data frame as my_df2.

h) Create the data frame my_df3 that is a subset of my_df2 containing only female entries.

i) Create the data frame my_df4 that is a subset of my_df2 containing only entries of males taller than 170.

Exercise 2

Create two data frames mydf1 and mydf2 as follows:

mydf1:

101	14

102	12
103	15
104	10

mydf2:

101	Paul
102	Helen
103	Emily
104	John

a) With column names: “id”, “age” for mydf1. and “id”, “name” for mydf2.

b) Merge mydf1 and mydf2 by their “id” # Look for help by google for how to do merging.

c) Give the merged data frames the new name mydf3.

d) Order mydf3 by decreasing age.