**##### EXTRA R EXERCISES – R FROM SCRATCH – R – LADIES #####**

R contains thousands of packages, each of them offering functions that do something useful. The package that is most commonly used (by far!) for data manipulation is called **dplyr** (pronounced *di-pl-a-yer*). Every package first has to be installed to your local machine (only once) and then loaded to your workspace (every time you open R). Let’s install and load dplyr.

**### DPLYR PACKAGE ###**

install.packages("dplyr")

library(dplyr)

**### LOADING DATA ###**

Now, before we practice using dplyr, let’s load the data we’re going to work with:

1. Read about one of the most popular R’s in-built datasets mtcars, using help() function or ?

2. Explore dataset’s structure and summary statistics. Pay attention to data types.

**?mtcars**

**data("mtcars")**

**str(mtcars)**

**summary(mtcars)**

**### DATA MANIPULATION ###**

Let’s learn about some of the most useful dplyr’s functions:

1. Find out about the following dplyr’s functions: mutate() and filter() (try looking up e.g. ?dplyr::mutate, etc.). What do they do? How do they work? What’s the syntax?
2. Create a new data.frame (new\_mtcars) that contains only observations where number of cylinders is greater or equal 6.
3. We’re going to create a new variable, cyl\_desc, that will put cyl in three buckets dependent on the value of cyl: low, medium and high (greater or equal 2, between 3 and 5 and greater or equal 6, respectively). For this reason let’s learn a bit about ifelse() or case\_when() statements (use R’s available help tools).
4. Create cyl\_desc using mutate().
5. Check the structure of new\_mtcars – what’s the class of the new variable? Turn it to factor.

**?dplyr::mutate**

**new\_mtcars <- filter(mtcars, cyl >=6)**

**new\_mtcars <- mutate(new\_mtcars,**

**cyl\_desc = ifelse(cyl <=2, "low",**

**ifelse(cyl >= 6, "high", "medium"))**

**)**

**str(new\_mtcars)**

**new\_mtcars$cyl\_desc <- as.factor(new\_mtcars$cyl\_desc)**

**### JOINING DATASETS ###**

It’s time to join the new dataset to the original mtcars. However, there are many ways to do it!

1. Read about available joins in dplyr (e.g. try ?dplyr::join)

2. Create a new dataset inner\_df where you use inner\_join() to join mtcars and new\_mtcars together.

3. Check the structure of inner\_df – how many variables and observations are there? What variables can you see? Are there any missing values?

4. Create a new dataset left\_df where you use left\_join() to join mtcars and new\_mtcars together.

5. Check the structure of left\_df – how many variables and observations are there? What variables can you see? Are there any missing values?

6. Create a new dataset right\_df where you use right\_join() to join mtcars and new\_mtcars together.

7. Check the structure of right\_df – how many variables and observations are there? What variables can you see? Are there any missing values?

8. Create a new dataset anti\_df where you use anti\_join() to join mtcars and new\_mtcars together.

9. Check the structure of anti\_df – how many variables and observations are there? What variables can you see? Are there any missing values?

**inner\_df <- inner\_join(mtcars, new\_mtcars)**

**str(inner\_df)**

**left\_df <- left\_join(mtcars, new\_mtcars)**

**str(left\_df)**

**summary(left\_df)**

**head(left\_df)**

**right\_df <- right\_join(mtcars, new\_mtcars)**

**str(right\_df)**

**?dplyr::anti\_join**

**anti\_df <- anti\_join(mtcars, new\_mtcars)**

**str(anti\_df)**