TUT SHEET 3

Note: There can be multiple ways of doing same thing in data frames.

INTRODUCTION TO DATA FRAMES:

The **dplyr** package in R is designed to make data manipulation tasks simpler and more intuitive than working with base R functions only.

A data frame is an R object that stores tabular data in a table structure made up of rows and columns. While data frames can be created in R, they are usually imported with data from a CSV, an Excel spreadsheet, or a SQL query.

Data frames have rows and columns. Each column has a name and stores the values of one variable. Each row contains a set of values, one from each column. The data stored in a data frame can be of many different types: numeric, character, logical, or NA.

LOADING AND READING A CSV:

To read data from a csv file and display it on console use **read.csv('filename')**.
One can even write data from a dataframe to a csv using **write.csv(df,'filename')**. It takes two arguments: df, representing the dataframe object and the name of the csv file that will hold the dataframe.

INSPECTING DATAFRAMES:

For larger dataframes it is not feasible to display the entire data all at once.
The **head()** function returns the first 6 rows of a data frame. If you want to see more rows, you can pass an additional argument **n** to head(). For example, head(df,8) will show the first 8 rows.
Similarly, **tail()** function returns last 6 rows of data frame. If you want to see more rows, you can pass an additional argument **n** to tail(). For example, head(df,8) will show the last 8 rows.

The function **summary()** will return summary statistics such as mean, median, minimum and maximum for each numeric column while providing class and length information for non-numeric columns.

PIPING:

The pipe operator, or **%>%,** helps increase the readability of data frame code by piping the value on its left into the first argument of the function that follows it.
Eg: df %>% head() is equivalent to head(df).
This operator is useful when there are multiple arguments for a function.

SELECTING COLUMNS:

Select the appropriate columns for your analysis using dplyr's **select()** function.
Select() takes a data frame as its first argument.

All additional arguments are the desired columns to select. It returns a new dataframe containing only the desired columns.
Eg: customers%>%select(age,gender) will select the columns age and gender from the dataframes customers.

FILTERING ROWS WITH LOGIC:
We can further manipulate a dataframe using **filter()** function and comparison operators.

MORE ON DATA FRAMES:

- To add more rows permanently to an existing dataframe, we need to bring in the new rows in the same structure as the existing data frame and use the rbind() function.
- To add more columns permanently to an existing dataframe, we need to bring in the new columns in the same structure as the existing data frame and use the cbind() function.
- You can change and access column and row names with **colnames()** and **rownames()** respectively.
  Example: colnames(dataframe_name) gives names of all columns.
  colnames(dataframe_name)[2] gives name of 2nd column.
  colnames(dataframe_name)->c(“”good”,”better”) sets names of column.
  Same goes for rownames().
- dim(datafrmae_name) returns no. of rows and columns of dataframe.
- Example of rows and column access:
  # Print out the values located on the first and second row, third column
  dataframe_name[1:2,3]

  # Print out the values located in the third column
  dataframe_name[,3]

  # Print out the values located in the third row
  dataframe_name[3,]
- Install packages :
  install.packages(“<package_name>”)
- To access the stored data, write data frame object name (“df”) with $ sign and name of the variable. That is,
- >df$V
- > df$V2
- >df[“V1”]
- > df[, 1]
- To access the data file which is not stored in the working directory, provide complete path of the file, such as
  > read.csv(“d:/Rdata/data.csv”)
- Remove Duplicate Rows based on all the variables (Complete Row)
  The **distinct function** is used to eliminate duplicates i.e if two rows are exactly same

one of them is removed.
x1 = distinct(dataframe_name)

- rename() : rename a variable name.
rename(dataframe_name,new_name=old_name)

EXERCISES

- If we have a data.frame df <- data.frame(a = c(1, 2, 3), b = c(4, 5, 6), c(7, 8, 9))...
    1. What is df[[1]]? (Test and find out)
    2. How do we select the c(4, 5, 6)?
    3. How do we select the 1?
    4. How do we select the 5?
    5. What is df[, 3]?
    6. What is df[1,]?
    7. What is df[2, 2]?