HW01 - Data Frame Basics

Stat 133, Spring 2018, Prof. Sanchez

Due date: Fri Feb-23 (before midnight)

The underlying purpose of the exercises in this assignment is to put in practice the basic manipulation operations for data frames, and other analytical tasks.

General Instructions

After completing the assignment, you should have the following file structure (10 pts). Notice that `first` and `last` correspond to your first and last names):

```
hw01/
   README.md
   imports-85.data
   imports-85-dictionary.md
   hw01-first-last.Rmd
   hw01-first-last.md
   hw01-first-last_files/
      figure-markdown_github/
          ...  # png files
```

- Write your narrative and code in an `Rmd` (R markdown) file.
- In the yaml header, set the `output` field as `output: github_document` (Do NOT use the default `"output: html_document"`).
- Name this file as `hw01-first-last.Rmd`, where `first` and `last` are your first and last names (e.g. `hw01-gaston-sanchez.Rmd`).
- Save the `Rmd` file in the folder `hw01/` of your (local) repository `hw-stat133`.
- Please do not use code chunk options such as: `echo = FALSE`, `eval = FALSE`, `results = 'hide'`. All chunks must be visible and evaluated.
- Include a `README.md` file for the directory `hw01/`.
- Use Git to *add* and *commit* the changes as you progress with your HW. Track changes in the `Rmd` and `md` files, as well as the generated folder and files containing the plot images.
- And don't forget to *push* your commits to your github repository; you should push the `Rmd` and `md` files, as well as the generated folder and files containing the plot images.
- Submit the link of your repository to bCourses. Do NOT submit any files (we will actually turn off the uploading files option).
- No work submitted to bCourses will be graded (no exceptions).
- No work emailed to the instructor or GSIs will be graded (no exceptions).
- If you have questions/problems, don't hesitate to ask us for help in OH or piazza.

- This assignment is not a warm-up assignment anymore; i.e. this HW does count towards your final grade.

Data Set

The data set for this assignment is the *Automobile Data Set* available in the UCI Machine Learning Repository:

https://archive.ics.uci.edu/ml/datasets/Automobile

The data description is provided in:

https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.names

And the data file is available in:

https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data

One possible way to download the data file—from R—is with the function `download.file()`. Here's how to do that from R's console:

```
# ------------------------------------------------------------------------
# do NOT include this code in your Rmd file (you'll lose points if you do)
# ------------------------------------------------------------------------

# assembling url (so the text fits on this pdf)
uci <- 'https://archive.ics.uci.edu/ml/machine-learning-databases'
autos <- '/autos/imports-85.data'
url <- paste0(uci, autos)

# download file to your working directory
download.file(url = url, destfile = 'imports-85.data')
```

1) Data Dictionary (10 pts)

When analysts work with data files that store information under some tabular format, such formats do not typically allow the inclusion of metadata (i.e. description about the data) in a convenient way. This is one of the main disadvantages of working with text files for data tables.

A major problem is that most analysts underestimate the value of metadata. To prevent that this happens to you, we are going to repeatedly request that you create data dictionaries for pretty much all HW assignments.

What do you include in a data dictionary? Typically we include a description of the data, the original source(s), who collected the data, and when it was collected. More important, we include the description of each variable (or field) specifying the measurement units (if any), and possibly the class of variable (e.g. quantitative, qualitative) and/or the ideal associated data type (e.g. integer, real, character, logical). Also, we can specify codification values, special meaning of some characters, or what value(s) represent missing values.

Based on the information in the file `imports-85.names`, you will have to create a data dictionary in a separate text file: e.g. `imports-85-dictionary.md`. or `imports-85-dictionary.txt`. The file extension `.md` indicates that the content of the dictionary is written in markdown syntax (don't confuse `.md` with `.Rmd`). By the way, don't use an Rmd file to write this dictionary.

2) Data Import (20 pts)

Because the data file `imports-85.data` is a CSV file, you can use the function `read.csv()` in base R, or the function `read_csv()` from the R package `"readr"`, to import the data in R. Write code to import the data with the following specifications:

- Create a character vector for the names of the columns using the names that appear in `imports-85.names`. If a given name contains a dash, then replace it with an underscore, e.g. `fuel_type`, `num_of_doors`, etc.
- Use the vector of names when invoking the functions to read the data. In other words, the imported data frame must contain column names.
- Non-numeric variables must be imported as `"character"` (not as factors).
- Numeric variables `curb_weight`, `engine_size`, `horsepower`, `peak_rpm`, `city_mpg`, `highway_mpg`, and `price` must be imported as `"integer"`.
- The rest of numeric variables must be imported as `"double"` or `"real"`.
- Following the specifications listed above, import the data using the `read.csv()` function, and also display the structure with `str()` (10 pts).
- Following the specifications listed above, import the data using the `read_csv()` function from the package `"readr"`, and also display the structure with `str()` (10 pts).

3) Technical Questions about importing data (10 pts)

Answer the following questions (using your own words). You do NOT need to include any commands.

a. If you don't provide a vector of column names, what happens to the column names of the imported data when you simply invoke `read.csv('imports-85.data')`?

b. If you don't provide a vector of column names, what happens to the column names of the imported data when you invoke `read.csv('imports-85.data', header = FALSE)`?

c. When using the reading table functions, if you don't specify how missing values are codified, what happens to the data type of those columns that contain `'?'`, e.g. `price` or `num_of_doors`?

d. Say you import `imports-85.data` in two different ways. In the first option you import the data without specifying the data type of each column. In the second option you do specify the data types. You may wonder whether both options return a data frame of the same memory size. You can actually use the function `object.size()` that provides an estimate of the memory that is being used to store an R object. Why is the data frame imported in the second option bigger (in terms of bytes) than the data frame imported in the first option?

e. Say the object `dat` is the data frame produced when importing `imports-85.data`. What happens to the data values if you convert `dat` as an R matrix?

4) Practice base plotting (10 pts)

As mentioned in class, the traditional or classic plotting approach of R has a heavy exploratory flavor. Functions like `hist()`, `boxplot()`, `barplot()`, `pie()`, `stars()`, and `pairs()`, produce graphics that are excellent for the initial phases of any exploratory data analysis (EDA) task. To be honest, their default settings are very minimalist, and sometimes a bit "ugly". But this does not mean that these graphics are useless. On the contrary, they let you know your data better. Keep in mind that most of the plots in these stages are to be consumed mostly by the data scientist (i.e. not to be part of your report).

Create the following plots—without using functions from the package `"ggplot2"`—and provide a concise description for each of them:

- histogram of `price` with colored bars.
- boxplot of `horsepower` in horizontal orientation.
- barplot of the frequencies of `body_style`, arranged in decreasing order.
- `stars()` plot of vehicles with turbo `aspiration`, using only variables `wheel-base`, `length`, `width`, `height`, and `price`.

5) Summaries (10 pts)

Use R code to answer the following questions:

a. What is the mean price of `fuel_type` gas cars? And what is the mean `price` of `fuel_type` diesel cars? (removing missing values)

b. What is the `make` of the car with twelve `num_of_cylinders`?

c. What is the `make` that has the most diesel cars?

d. What is the `price` of the car with the largest amount of `horsepower`?

e. What is the bottom 10th percentile of `city_mpg`?

f. What is the top 10th percentile of `highway_mpg`?

g. What is the median price of those cars in the bottom 10th percentile of `city_mpg`?

6) Technical Questions about data frames (10 pts)

Answer the following questions (using your own words). You do NOT need to include any commands.

a. What happens when you use the dollar `$` operator on a data frame, attempting to use the name of a column that does not exist? For example: `dat$xyz` where there is no column named `xyz`.

b. Which of the following commands fails to return the vector `mpg` which is a column in the built-in data frame `mtcars`:

   1. `mtcars$mpg`
   2. `mtcars[ ,1]`
   3. `mtcars[[1]]`
   4. `mtcars[ ,mpg]`
   5. `mtcars[["mpg"]]`
   6. `mtcars$"mpg"`
   7. `mtcars[ ,"mpg"]`

c. Based on your answer for part (b), what is the reason that makes such command to fail?

d. Can you include an R `list` as a "column" of a data frame? YES or NO, and why.

e. What happens when you apply `as.list()` to a data frame? e.g. `as.list(mtcars)`

f. Consider the command: `abc <- as.list(mtcars)`. What function(s) can you use to convert the object `abc` into a data frame?

7) Correlations of quantitative variables (10 pts)

Except for `symboling` and `normalized_losses`, use the rest of the quantitative variables (both integer and real) to compute a matrix of correlations between such variables. See how to use the function `na.omit()` to create a new data frame with the quantitative variables, that does not contain missing values. Call this data frame `qdat`. *Hint:* see the function `cor()`.

Read the post *Correlograms* by Xia Liu, available in the file `correlograms-xia-liu`, inside the folder `papers` of the course github repo:

https://github.com/ucb-stat133/stat133-spring-2018/blob/master/papers/correlograms-xia-liu.pdf

Based on the matrix of correlations between the quantitative variables, plot two correlograms, and comment on the patterns and values that you observe.

8) Principal Components Analysis (20 pts)

Read the tutorial on Principal Components Analysis (PCA) availabe in the github repository

https://github.com/ucb-stat133/stat133-spring-2018/blob/master/tutorials/06-principal-components.md

8.1) Run PCA (10 pts)

- Use `prcomp()` to perform a principal components analysis on `qdat`; use the argument `scale. = TRUE` to carry out PCA on standardized data.
- Examine the eigenvalues and determine the proportion of variation that is "captured" by the first three components.

8.2) PCA plot of vehicles, and PCA plot of variables (10 pts)

- Use the first two components to graph a scatterplot of the vehicles (do not use `"ggplot2"` functions).
- Use the first two loadings (i.e. eigenvectors) to graph the variables.
- Optionally, you can call `biplot()` of the `"prcomop"` object to get a simultaneous plot of both the vehicles and the variables.
- Based on the previous plots, provide a concise description of the patterns that you observe.