Problem Set 1

In this first problem set you will get a first exposure to advanced R tasks. Your solution should be composed of a well-structured R script which should provide the designated functions. Besides the functions the code should be directly runnable or at least sufficiently well documented (working directoy, path settings) to be executed.

Furthermore, provide a documentation (in text format or powerpoint) where you document and illustrate your general approach and document the qualitative tasks. Grading will reflect your performance on both the coding as well as the documentation and interpretation tasks – however, there is no fixed grading scheme between the two categories.

This problem set is **due on May 2nd by 12.00** through the wuecampus group functionality. If you have not chosen a group yet do so as soon as possible and communicate the choice to the course instructor.

Cooperation with other groups is not permitted and will lead to severe credit deductions.

1. You are helping a regional car dealer's marketing department. You are required to program an R function which automatically generates banners like the following. Proceed along the steps to accomplish this task.

```
*************************
* Toyota Corolla        *
* Horsepower: 47        *
* Cylinders: 4          *
* Fuel Efficiency: 34mpg *
* 1/4 mile time: 19sec  *
*************************
```

   a. Create a function which creates unformatted output of a single row data.frame equivalent to the internal data set *mtcars*.
   createAd(vehicleData)
   b. Using sprintf or paste, length and repeat format your output analogue to the above.
   createFormattedAd(vehicleData)
   c. To improve marketing chances, the vendor wants to include a relative ranking if the fuel efficiency rating, horsepower or quarter mile time is in the top 10% of the data set. In these cases, include "(Top x%)" behind the corresponding entry.
   createFormattedAdWithComparisons(vehicleData)
   d. To automate the campaign, expand your function to take a data.frame with multiple cars plus an additional integer argument *n* which specifies the number of ads that should be created. Then randomly sample n rows and create the ads for these vehicles. Watch out not to create the same ad twice
   createFormattedAdsWithComparisons (vehiclesData, n)
   e. For the purpose of car sales, the data set is clearly missing two essential data points – price and mileage. Combine the provided data files carMileage.csv and carPrices.csv with mtcars and include unformatted price and mileage statements in your Ad function. Create a vector or list with all the ads of cars in this expanded mtcars.

2. You are working in purchasing for a big manufacturing firm. To create a base for data analysis you are expected to create a combined data model from manufacturing and purchasing data. You are provided with the following data files:
    - quotes.csv: This file contains information on price quotes from our suppliers. Prices can be quoted in 2 ways: bracket and non-bracket pricing. Bracket pricing has multiple levels of purchase based on quantity (in other words, the cost is given assuming a purchase of quantity tubes). Non-bracket pricing has a minimum order amount (min_order) for which the price would apply. Each quote is issued with an annual_usage, an estimate of how many tube assemblies will be purchased in a given year.
    - assemblies.csv: This file contains information on assemblies, which are made of multiple parts. The main piece has a specific diameter, wall thickness, length, number of bends and bend radius. Either end of the assembly (End A or End X) typically has some form of end connection allowing the assembly to attach to other features. Special tooling is typically required for short end straight lengths (end_a_1x, end_a_2x refer to if the end length is less than 1 times or 2 times the diameter, respectively). Other components can be permanently attached to an assembly such as bosses, brackets or other custom features.
    - bom.csv: This file contains the list of components, and their quantities, used on each assembly.
    - steps.csv: This file contains the list of unique steps taken for an assembly. These can refer to materials, processes, rust protection, etc.
    - end_form.csv: Some end types are physically formed utilizing only the wall of the assembly. These are listed here.
    - components.csv: This file provides extra information on the components.

    a. Set up an infrastructure for reading in the files. Provide an executive summary of the data set (qualitative with respect to the data structure as well as quantitative using suitable summary statistics).
    b. To obtain more comprehensive information on the product designs combine assembly data set with the bill of materials and the steps information.
    c. We now want to focus on specs. Your data from b) will most likely have many **specs** columns in wide data format. Change it to a long format with a single id variable tube_assembly_id and measure ids corresponding to the spec columns. How can we assess an assembly's complexity with respect to its specs from this data?
    d. We now want to go back to wide format with the data from c: However, we are no longer interested in the distinction of the spec number. Drop the column (e.g., data$variable <- NULL) and cast the data frame to wide format. Interpret your result.
    e. Combine all the data sets to a single (wide) data frame (table) [do not use the data from b / c / d]. What are the dimensions of the object? Which columns are redundant? Do you see a problem with the component columns concerning analysis of pricing patterns?
    f. To harmonize the bill of materials data, you are provided the following code where bom is the bill of materials data frame and X can take the values 1 to 8.

```
bomWX <- dcast(bom, tube_assembly_id~component_id_X,
          value.var="quantity_1",fill=0)
```

    What is a bomWX object? Interpret the dimensions of the results bomW1 and bomW8.

    [BONUS: `rowSums(bomWideX[,-c(1)] != 0)` allows you to count rowwise the number of non-zero entries]

3. The following call will provide you with data from google's book API.

```
library(RCurl)
library(RJSONIO)
URL =
"https://www.googleapis.com/books/v1/volumes?q=george+r+r+martin&
maxResults=40"
response_parsed <- fromJSON(getURL(URL,ssl.verifyhost = 0L,
ssl.verifypeer = 0L))
```

   a. Describe the structure of the response object. Explain dimensions and nesting of the elements.
   b. Using *apply calls extract the author, the title, publishing date and the rating of each book in the response. [You will need minimal functions which do the addressing – these can be defined within the *apply call!]
   c. Combine your individual calls in one function which specifies how many items to be shown. Sort the list by date and title and return it.
      getBookList(numberOfItems)
   d. Create a function which provided with a string argument specifying a book id (from the 40 books in your list) returns where this book is available as well as price and a buy link. [You may want to change the API call to simplify and generalize this task.]
      getBookSalesInfo(response)