Homework 1

Lan Huong Nguyen

October 11, 2017

Exercise 1: R Basics

a. Arithmetic operations [5pt]

Compute the following using R:

- $1.46 \log_2(12!)$
- $4.02 \sqrt[3]{7^2 + e^7}$
- $20 \cos(2\pi + 0.25) + 32 \sin\left(\frac{3\pi}{4}\right)$
- $\left\lfloor \frac{4.011\pi}{3} \binom{5}{2} \right\rfloor$
- A sum of numbers from 3 to 9

b. Matrix operations [5pt]

Generate a matrix A with 10 rows and 6 columns with entries being random uniform numbers on an interval $[0, 1]$. Then generate a matrix B with 6 rows and 5 columns where entries are random integers in teh interval 10 to 30 (inclusive). Use `set.seed()` function with a chosen seed (recored the seed) for reproducibility. Type in `?set.seed` in the R console to learn more about the function. With the two matrices compute:

- AB (a matrix product)
- sum of the 3rd row of A and 4th column of B
- sum of all entries of A multiplied by the element of B in the 5th row and 4th column.

c. Data-frames [5pt]

Create a data-frame, 'birthdays', which stores information on the birthdays of 6 people your know, i.e. your friends or family members. The data-frame should have columns:

1. 'first': first name
2. 'last': last name
3. 'birthday': the person's birthday in formapt YYYY-MM-DD ("%Y-%m-%d")
4. 'relationship': "friend", "relative", "mom", "dad", "sister", "brother", etc.
5. 'YOR': years of relationship (how long have you known the person)
6. 'city': city where the person lives

Convert the birthdays to date objects using `as.Date()` function. Compute the difference (in days) between your birthday and the birthday of each of the people and apppend that information as a new column 'bday_diff' of the data-frame.

d. Factors [5pt]

In this part of the exercise we use a built-in data set on student's sleep data. Learn more about this dataset by reading the help page, acessed by running `?sleep` in the R console. Note that in this dataset the patient ID is a factor with lavels 1 through 10. Rename the patient's IDs to be letters of the alphabet, with "A" corresponding to 10, "B" to 9, "C" to 8, ..., and "J" to 1.

Exercise 2: Functions

a. Parametric function [10pt].

- Write a function in are that evaluates the following:

$$f(\theta) = 7 - 0.5\sin(\theta) + 2.5\sin(3\theta) + 2\sin(5\theta) - 1.7\sin(7\theta) + \\ + 3\cos(2\theta) - 2\cos(4\theta) - 0.4\cos(16\theta)$$

- Generate a vector, `theta`, equal to a sequence from 0 to 2π with increments of 0.01
- Compute a vector $x = f(\theta) \cdot \cos(\theta)$ and $y = f(\theta) \cdot \sin(\theta)$ for θ you just created.
- Plot a scatter plot of (x, y) with two vectors computed.

b. Multiple arguments [10pt]

Write a function `time_diff()` that takes two dates as inputs and returns the difference between them in units of "hours", "days", "months" or "years", defined by an optional argument 'units', set by default to "days". Use the function to compute the number of months, days, and hours left to 2018 FIFA World Cup opening game in "2018/06/14"

Exercise 3: Control Flow

a. Fibonacci numbers [5pt].

Fibonacci sequence starts with numbers 1 and 2, and each subsequent term is generated by adding the previous two terms. The first 10 terms of the Fibonacci sequence are thus: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... Find the total sum of **even numbers**, not exceeding one million which also belong to the Fibonacci sequence.

b. Prime numbers [15pt].

- Write a function `is_prime(n)` that returns `TRUE` if n is prime and `FALSE` otherwise. Note that apart from 2 and 3, all primes can be represented as $6k \pm 1$ where k is some integer, this does NOT meant that all numbers of the form $6k \pm 1$ are prime. You can use this information to highly reduce your computation costs and speed up your function.
- Write a function that takes a number as an input and returns all primes less than or equal to that number. Use the function to compute the sum of all primes not greater than 10,000.
- Write a function that takes an integer K as na input and returns the first K primes. The, compute the sum of 1000 first primes using your function.

If you don't know where to begin from, then read the wikipedia entry on primality test.

Exercise 4: Data Import/Export

a. Import data [10pt]

Go to the following URL: https://raw.githubusercontent.com/cme195/cme195.github.io/master/assets/data/share-of-people-who-say-they-are-happy.txt

Note the structure and format of the data. Then, use the function `read.table()` with suitable arguments to import the data. Use R to find the cuntry with the highest share of "happy" people in 2014.

b. Export data [10pt]

Select the observations from the data set on happiness which were reported after year 2000. Export the subset of the data as a tab-text file to a chosen location on your computer.

Exercise 5. Data manipulations

a. `dplyr` functions [10pt]

Install a package `nycflights13` to acess datasets on filights and airports in the city of New York in 2013. Print the dataset 'fights' and note that it is a tibble, and it has 336,776 observations! You can learn more about the dataset using `?flights` from the console after loading the `nycflights13` package. Use `dplyr` functions (and the `%>%` operator) to find for each departure airport, 'origin', the carrier with the longest average departure delay, 'dep_delay'. For the carriers found (for each origin airport) report both the average arrival and departure delay. **Note**: Since, the dataset contatins missiing values, when computing the averages, remember to exclude the missing values (use 'na.rm = TRUE' in `mean()` function).

b. Joining/merging datasets [10pt]

The `nycflights13` package also includes a data-frame, '`airlines`', which includes the full names of the carries. First, select columns: year, month, day, time_hour, dep_time, arr_time, carrier, flight, tailnum, origin, dest from the dataset '`flights`' and save it as a data-frame '`flights2`'.

Add a column with the full name of the carrier to '`flights2`' by merging with the '`airline`' data frame. Note which columns where used for merging.

Another dataset available in the package is '`weather`' which stores data on weather at different airports at specific days and times. Use this dataset to merge the weather information to the data-frame obtained in the previous step. Note which columns did the joining function use to merge the two data tables.

There, is also a data-frame '`planes`' included in the package. '`planes`' share columns 'year' and 'tailnum' in '`planes`' with '`flights2`' data-frame, but column 'year' in '`planes`' means a different thing (year produced) than in '`flights2`'. Use only the column 'tailnum' to merge '`flights2`' and '`planes`'.

Now, use the data-frame '`airports`' to merge '`flights2`' with the information on the origin airport. Note that the column 'faa' is the airport identifier in the dataset '`airports`'. You must use the `by =` argument in the join function and specify which columns from '`flights2`' you are matching to which column in '`airports`'.