Lab3-Solutions

Claire Donnat

7/2/2020

Exercise 1: R Basics

Material: Week 1 Session 2

a. Arithmetic operations

Compute the following using R:

- $4.84 \log_{10}(51!)$, where $x!$ is factorial of x
- $4.02 \sqrt[3]{7^2 + e^7}$
- $20 \cos(2\pi + 0.25) + 32 \sin\left(\frac{3\pi}{4}\right)$
- $\left\lfloor \frac{4.011\pi}{3} \binom{5}{2} \right\rfloor$, where $\lfloor x \rfloor$ means rounding to the largest integer not greater than x. where $\binom{x}{y}$ is the notation for combination
- $8.1 \sum_{i=1}^{100} \frac{1}{i}$

```
4.84 * log10(factorial(51))
```

```
## [1] 320.3627
```

```
4.02 * (7^2 + exp(7))^{1/3}
```

```
## [1] 42.06374
```

```
20 * cos(2*pi + 0.25) + 32 * sin(3 * pi / 4)
```

```
## [1] 42.00567
```

```
floor((4.011 * pi/3) * choose(5,2))
```

```
## [1] 42
```

```
8.1 *sum(1/seq_len(100))
```

```
## [1] 42.01776
```

b. Matrix operation

Generate a matrix A with 15 rows and 5 columns with entries being random uniform numbers on an interval $[0, 1]$. Then generate a matrix B with 5 rows and 7 columns where entries drawn from a Gaussian distribution with mean 0 and variance 10. Use `set.seed()` function with a chosen seed (record 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)

- multiply the 3rd row of A by the 4th column of B and compute the sum of entries in the resulting vector, then check that agrees with a corresponding term in the matrix product you computed in the previous part.
- obtain a vector which is a product of matrix multiplication between in matrix A and the 4th column of B.

```
set.seed(123)
(A <- matrix(runif(15*5), nrow = 15))
```

```
##            [,1]       [,2]       [,3]       [,4]         [,5]
##  [1,] 0.2875775 0.89982497 0.96302423 0.13880606 0.6651151946
##  [2,] 0.7883051 0.24608773 0.90229905 0.23303410 0.0948406609
##  [3,] 0.4089769 0.04205953 0.69070528 0.46596245 0.3839696378
##  [4,] 0.8830174 0.32792072 0.79546742 0.26597264 0.2743836446
##  [5,] 0.9404673 0.95450365 0.02461368 0.85782772 0.8146400389
##  [6,] 0.0455565 0.88953932 0.47779597 0.04583117 0.4485163414
##  [7,] 0.5281055 0.69280341 0.75845954 0.44220007 0.8100643530
##  [8,] 0.8924190 0.64050681 0.21640794 0.79892485 0.8123895095
##  [9,] 0.5514350 0.99426978 0.31818101 0.12189926 0.7943423211
## [10,] 0.4566147 0.65570580 0.23162579 0.56094798 0.4398316876
## [11,] 0.9568333 0.70853047 0.14280002 0.20653139 0.7544751586
## [12,] 0.4533342 0.54406602 0.41454634 0.12753165 0.6292211316
## [13,] 0.6775706 0.59414202 0.41372433 0.75330786 0.7101824014
## [14,] 0.5726334 0.28915974 0.36884545 0.89504536 0.0006247733
## [15,] 0.1029247 0.14711365 0.15244475 0.37446278 0.4753165741
```

```
(B <- matrix(rnorm(5*7), nrow = 5))
```

```
##            [,1]       [,2]        [,3]       [,4]       [,5]       [,6]
## [1,] -0.7717918 -0.1639310 -0.88643672  1.2283928 -1.0700682  2.1499193
## [2,]  0.2865486  1.2429188 -1.31893760  0.2760235  1.6858872 -1.3343536
## [3,] -1.2205120 -0.9343851  0.02884391 -1.0489755 -0.2416898  0.4958705
## [4,]  0.4345504  0.3937087 -0.43212979 -0.5208693 -0.4682005  1.2339762
## [5,]  0.8001769  0.4036315  1.68987252  1.6232025 -0.7729782  0.6343621
##            [,7]
## [1,]  0.4120223
## [2,]  0.7935853
## [3,] -0.1524106
## [4,] -0.2288958
## [5,] -0.9007918
```

```
A %*% B
```

```
##              [,1]       [,2]        [,3]       [,4]        [,5]       [,6]
##  [1,] -0.54696100  0.4945416 -0.34997723  0.59875898  0.39741414  0.4883263
##  [2,] -1.46200377 -0.5364271 -0.93776350  0.12235002 -0.82915700  2.1615697
##  [3,] -0.63687921 -0.3217164  0.04942006  0.17001573 -1.04862599  1.9842078
##  [4,] -1.22328391 -0.2649823 -0.84356295  0.64762342 -0.92092895  2.3575653
##  [5,]  0.54225092  1.6757487 -1.08594063  2.26841910 -0.43446187  2.3358044
##  [6,]  0.01538873  0.8507910 -0.46171788  0.50445955  0.96728449 -0.5110159
##  [7,] -0.29442164  0.5668977 -0.18220196  1.12891554 -0.41363278  1.6465757
##  [8,]  0.22787454  1.0900441 -0.60202400  1.94856665 -0.92944698  2.6724801
##  [9,]  0.15955556  1.2167101 -0.50135350  1.84394138  0.33816994  0.6709314
## [10,]  0.14848131  0.9220881 -0.76205696  0.92067796 -0.04176089  1.1928069
## [11,] -0.01627499  0.9762041 -0.59284231  2.33823460 -0.54377876  1.9159599
## [12,] -0.14102988  0.5187522 -0.09929077  1.22712445 -0.21414073  1.0107418
```

```
## [13,]  0.03797335  0.8240534 -0.49773536  1.32272823 -0.72503793  2.2491572
## [14,] -0.41983350  0.2735253 -1.26406796 -0.06886378 -0.63395587  2.1330349
## [15,]  0.31971841  0.3628176  0.36053506  0.58361683 -0.44169676  0.8641720
##              [,7]
##  [1,]  0.05489868
##  [2,]  0.24379870
##  [3,] -0.35591894
##  [4,]  0.19477566
##  [5,]  0.21104795
##  [6,]  0.23736406
##  [7,] -0.17912450
##  [8,] -0.07165387
##  [9,]  0.22430799
## [10,]  0.14859630
## [11,]  0.20785261
## [12,] -0.04062336
## [13,] -0.12453488
## [14,]  0.20375974
## [15,] -0.37795392
```

```
sum(A[3, ] * B[, 4])
```

```
## [1] 0.1700157
```

```
A %*% B[, 4]
```

```
##              [,1]
##  [1,]  0.59875898
##  [2,]  0.12235002
##  [3,]  0.17001573
##  [4,]  0.64762342
##  [5,]  2.26841910
##  [6,]  0.50445955
##  [7,]  1.12891554
##  [8,]  1.94856665
##  [9,]  1.84394138
## [10,]  0.92067796
## [11,]  2.33823460
## [12,]  1.22712445
## [13,]  1.32272823
## [14,] -0.06886378
## [15,]  0.58361683
```

c. Factors

In this part of the exercise we use a built-in data set, `sleep`, on student's sleep. This data stores information on the increase in hours of sleep, type of drug administered, and the patient ID, in respective columns. You can learn more about this dataset from the page, accessed by calling `?sleep`.

The column `ID` for patient identity is a factor with labels from 1 to 10. Rename the ID labels to letters of the alphabet in the reverse order, with label "A" assigned to patient 10, "B" to patient 9, "C" to patient 8, . . . , and "J" to patient 1.

```
head(sleep)
```

```
##   extra group ID
```

```
## 1   0.7     1  1
## 2  -1.6     1  2
## 3  -0.2     1  3
## 4  -1.2     1  4
## 5  -0.1     1  5
## 6   3.4     1  6
```

```r
sleep$NewID = factor(sleep$ID, levels = seq(10, 1), labels = LETTERS[seq_len(10)])
```

d. Data Frame

Create a data frame, 'birthdays', which stores information on the birthdays of 5 people either real of fictional. The data table should have columns:

1. 'first': first name
2. 'last': last name
3. 'birthday': the person's birthday in format YYYY-MM-DD ("%Y-%m-%d")
4. '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 append that information as a new column 'bday_diff' of the data-frame.

```r
birthdays <- data.frame(
    first = c("Harry", "Ron", "Hermione", "Ginny", "Draco"),
    last = c("Potter", "Weasley", "Granger", "Weasley", "Malfoy"),
    birthday = c("1980-07-31", "1980-03-01", "1979-09-19", "1981-08-11",
                 "1980-06-05"),
    city = c("Hogwarts", "Hogwarts", "Hogwarts", "Hogwarts", "Hogwarts")
    )

birthdays$birthday = as.Date(birthdays$birthday)
birthdays$bday_diff = as.Date(birthdays$birthday) - as.Date("1926-12-31")
birthdays
```

```
##      first    last   birthday     city  bday_diff
## 1    Harry  Potter 1980-07-31 Hogwarts 19571 days
## 2      Ron Weasley 1980-03-01 Hogwarts 19419 days
## 3 Hermione Granger 1979-09-19 Hogwarts 19255 days
## 4    Ginny Weasley 1981-08-11 Hogwarts 19947 days
## 5    Draco  Malfoy 1980-06-05 Hogwarts 19515 days
```

Exercise 2: Programming

a. Parametric function

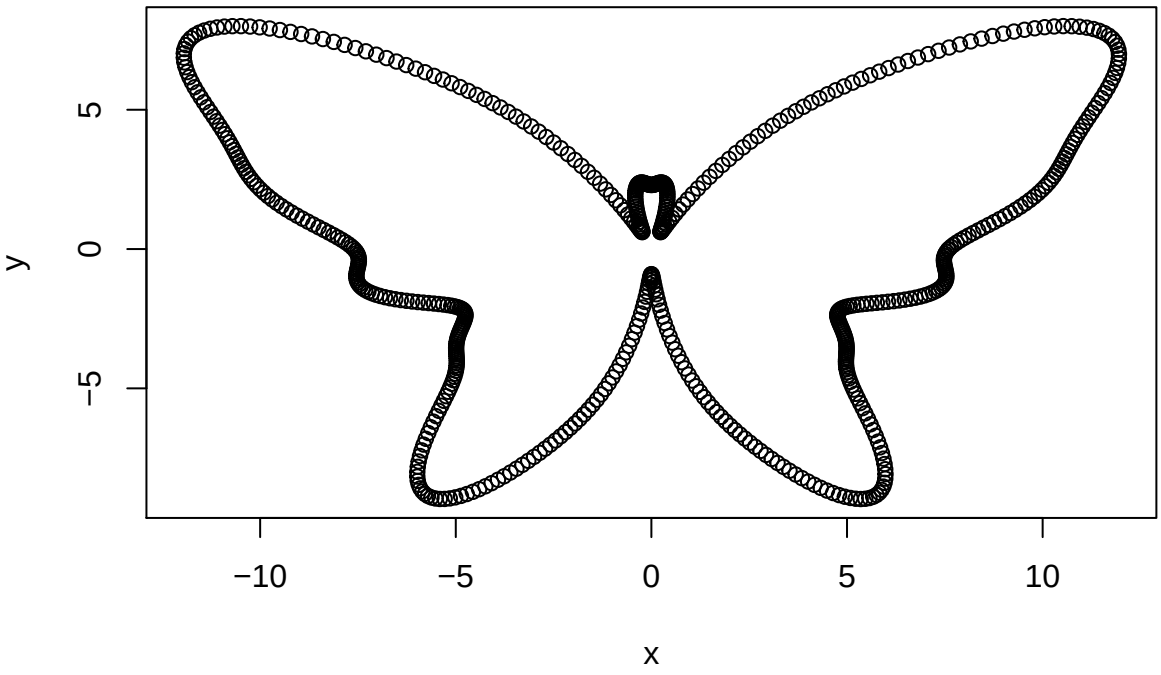
- Write a function in are that evaluates the following:

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

- 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.

```
f <- function(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)
}
theta <- seq(0, 2*pi, by = 0.01)
x <- f(theta) * cos(theta)
y <- f(theta) * sin(theta)
plot(x, y)
```



b. Multiple arguments

Write a function `time_diff()` that takes two dates as inputs and returns the difference between them in units of "hours", "days", "weeks", "months", or "years", defined by an optional argument 'units', set by default to "days". Use the function to compute time left to your next birthday separately in units of: months, days, and hours.

```
mybday = "2019-08-01"
time_diff <- function(d1, d2, units = "days") {
    time_diff <- as.numeric(as.Date(d1) - as.Date(d2))
    switch(
        units,
        "hours" = time_diff * 24,
        "days" = time_diff,
        "weeks" = time_diff / 7,
        "months" = round(time_diff / (365.25/12)),
        "years" = round(time_diff / 365.25)
    )
}
```

```
t_h <- time_diff(as.Date(mybday), Sys.Date(), "hours")
t_days <- time_diff(as.Date(mybday), Sys.Date())
t_months <- time_diff(as.Date(mybday), Sys.Date(), "months")


cat("There are", t_h , "hours left to my next birthday.\n")
```

```
## There are -8040 hours left to my next birthday.
```

```
cat("There are", t_days , "days left to my next birthday.\n")
```

```
## There are -335 days left to my next birthday.
```

```
cat("There are", t_months , "months left to my next birthday.\n")
```

```
## There are -11 months left to my next birthday.
```

There are 11 months

c. Control flow: Fibonacci numbers

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** in the Fibonacci sequence, each not exceeding one million.

```
prev_fib <- 1; fib <- 2;
even_fib_sum <- fib
while(fib <= 1e6) {
    new_fib <- prev_fib + fib
    prev_fib <- fib
    fib <- new_fib
    if (fib %% 2 == 0) {
        even_fib_sum <- even_fib_sum + fib
    }
}
even_fib_sum
```

```
## [1] 1089154
```

Exercise 3: Data Import/Export and transformation

Material: Week 1 Session 3.

a. Import data

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

Observe the structure and format of the data. Then, use a function from `readr` package to read the data in the URL into R. Then, find the country with the highest share of happy people in 2014.

```
library(readr)
url <- "https://raw.githubusercontent.com/cme195/cme195.github.io/master/assets/data/share-of-people-wh
happy <- read_delim(url, ";")
```

```
## Parsed with column specification:
## cols(
##   Entity = col_character(),
##   Code = col_character(),
##   Year = col_double(),
##   ShareOfHappyPeople = col_double()
## )
```

```
ind_max = which.max(happy$ShareOfHappyPeople[which(happy$Year==2014)])
print(happy[which(happy$Year==2014)[ind_max],])
```

```
## # A tibble: 1 x 4
##   Entity Code   Year ShareOfHappyPeople
##   <chr>  <chr> <dbl>              <dbl>
## 1 Qatar  QAT    2014               98.1
```

b. Export data

Filter observations from the data set on happiness that correspond to years after 2000. Export the subset of the data as a tab-delimited text file to a chosen location on your computer.

```
write_tsv(happy[which(happy$Year > 2000),], "../path/to/chosen/destination/file.tsv")
```