Lab 2 Sample Solution

Zelin (James) Li

6/29/2020

Exercise 0: Swirl courses (skipped)

Exercise 1: Vectors

1. Generate and print a vector of 10 random numbers between 5 and 500.

```
randomNumber <- sample(5:500, 10)
print(randomNumber)
```

```
##  [1] 141  82 481 229 390 328 408 168 314 493
```

2. Generate a random vector Z of 1000 letters (from "a" to "z"). Hint: the variable `letters` is already defined in R.

```
Z <- sample(letters, 1000, replace = TRUE)
```

3. Print a summary of Z in the form of a frequency table.

```
table(Z)
```

```
## Z
##  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
## 34 30 39 36 35 31 34 42 39 41 38 37 44 42 43 34 47 31 39 38 45 48 39 26 45 43
```

4. Print the list of letters that appear an even number of times in Z.

```
table(Z)[table(Z) %% 2 == 0]
```

```
## Z
##  a  b  d  g  h  k  m  n  p  t  v  x
## 34 30 36 34 42 38 44 42 34 38 48 26
```

Exercise 2: Matrices

1. Create the following 5 by 5 matrix and store it as variable X.

```
X <- matrix(1:25, nrow = 5, byrow = FALSE)
```

2. Create a matrix Y by adding an independent Gaussian noise (random numbers) with mean 0 and standard deviation 1 to each entry of X.

```
Y <- X + matrix(rnorm(25), nrow = 5, byrow = FALSE)
Y
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 2.4878618 6.116521 12.28478 14.91132 21.96812
## [2,] 1.2232820 6.536176 11.17276 15.79270 21.35402
```

```
## [3,] 0.9083492 7.317506 12.45487 17.72527 23.87082
## [4,] 2.8354373 9.635730 14.21332 19.18398 23.56368
## [5,] 5.1119372 7.703597 15.80114 17.57066 25.27014
```

3. Find the inverse of Y.

```
invY <- solve(Y)
invY
```

```
##            [,1]       [,2]      [,3]       [,4]       [,5]
## [1,] -0.2901485  2.8590517 -2.338429 -0.0608509  0.1019324
## [2,]  2.5989254  0.7305438 -2.393196  1.0545731 -1.5993442
## [3,] -0.6852318 -4.2789946  3.809664 -0.1790191  0.7797967
## [4,] -2.6039606  0.5363436  1.191770 -0.6417963  1.2831582
## [5,]  1.5054488  1.5016129 -2.008184  0.2490113 -0.8732830
```

4. Show numerically that the matrix product of Y and its inverse is the identity matrix.

```
product <- Y %*% invY
product
```

```
##              [,1]          [,2] [,3]          [,4]         [,5]
## [1,] 1.000000e+00  0.000000e+00    0  8.881784e-16 3.552714e-15
## [2,] 0.000000e+00  1.000000e+00    0 -3.552714e-15 0.000000e+00
## [3,] 0.000000e+00  0.000000e+00    1 -8.881784e-16 3.552714e-15
## [4,] 7.105427e-15 -2.131628e-14    0  1.000000e+00 0.000000e+00
## [5,] 0.000000e+00  7.105427e-15    0  0.000000e+00 1.000000e+00
```

```
round(product, digits = 2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

Exercise 3: Data Frames

1. Create the following data frame and name it "exams".

```
set.seed(123)
exams <- data.frame(
  student = c("Alice", "Sarah", "Harry", "Ron", "Kate"),
  score = sample(80:100, 5),
  letter = sample(c("A","B"), 5, replace = TRUE),
  late = sample(c(T, F), 5, replace = TRUE)
)
```

2. Compute the mean score for this exam and print it.

```
mean(exams$score)
```

```
## [1] 91.2
```

3. Find the student with the highest score and print the corresponding row of "exams". Hint: use the function `which.max()`.

```
exams[which.max(exams$score),]
```

```
##   student score letter  late
## 2   Sarah    98      B FALSE
```

Exercise 4: Control Flow

Part 1

a. Print all the letters of the Latin alphabet

```
for (x in letters){
  print(x)
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
## [1] "e"
## [1] "f"
## [1] "g"
## [1] "h"
## [1] "i"
## [1] "j"
## [1] "k"
## [1] "l"
## [1] "m"
## [1] "n"
## [1] "o"
## [1] "p"
## [1] "q"
## [1] "r"
## [1] "s"
## [1] "t"
## [1] "u"
## [1] "v"
## [1] "w"
## [1] "x"
## [1] "y"
## [1] "z"
```

b. Print the numbers 10 to 100 that are divisible by 7

```
for (x in 10:100){
  if (x %%7 == 0){
    print(x)
  }
}
```

```
## [1] 14
## [1] 21
## [1] 28
## [1] 35
## [1] 42
## [1] 49
```

```
## [1] 56
## [1] 63
## [1] 70
## [1] 77
## [1] 84
## [1] 91
## [1] 98
```

c. Print the numbers 1 to 100 that are divisible by 5 but not by 3.

```
for (x in 1:100){
  if (x %% 5 == 0 & x %% 3 != 0){
    print(x)
  }
}
```

```
## [1] 5
## [1] 10
## [1] 20
## [1] 25
## [1] 35
## [1] 40
## [1] 50
## [1] 55
## [1] 65
## [1] 70
## [1] 80
## [1] 85
## [1] 95
## [1] 100
```

Part 2

a. Find all numbers not greater than 10,000 that are divisible by 5, 7 and 11 and print them.

```
x <- 1
while (x <= 10000){
  if ((x %% 5 == 0)& (x %% 7 == 0) & (x %% 11 == 0)){
    print(x)
  }
  x <- x+1
}
```

```
## [1] 385
## [1] 770
## [1] 1155
## [1] 1540
## [1] 1925
## [1] 2310
## [1] 2695
## [1] 3080
## [1] 3465
## [1] 3850
## [1] 4235
## [1] 4620
## [1] 5005
## [1] 5390
```

```
## [1] 5775
## [1] 6160
## [1] 6545
## [1] 6930
## [1] 7315
## [1] 7700
## [1] 8085
## [1] 8470
## [1] 8855
## [1] 9240
## [1] 9625
```

b. Print for each of the numbers x = 2, . . . 20, all numbers that divide x (all factors) excluding 1 and x. Hence, for 18, it should print 2 3 6 9.

```
for (x in 2:20){
  vec <- c()
  for (i in 2:(x-1)){
    if (x == 2) next
    if (x %% i == 0){
      vec <- c(vec, i)
    }
  }
  cat("Factors of", x, "are:", vec, "\n")
}
```

```
## Factors of 2 are:
## Factors of 3 are:
## Factors of 4 are: 2
## Factors of 5 are:
## Factors of 6 are: 2 3
## Factors of 7 are:
## Factors of 8 are: 2 4
## Factors of 9 are: 3
## Factors of 10 are: 2 5
## Factors of 11 are:
## Factors of 12 are: 2 3 4 6
## Factors of 13 are:
## Factors of 14 are: 2 7
## Factors of 15 are: 3 5
## Factors of 16 are: 2 4 8
## Factors of 17 are:
## Factors of 18 are: 2 3 6 9
## Factors of 19 are:
## Factors of 20 are: 2 4 5 10
```

Exercise 5: Functions

Part 1

a. Create a function what will return the number of times a given integer is contained a given vector of integers. The function should have two arguments one for a vector and the other for a scalar.

```r
no_of_int <- function(vec, int){
  count <- 0
  for (x in vec){
    if (x == int){
      count <- count + 1
    }
  }
  return (count)
}

no_of_int_fast <- function(vec, int){
  return (sum(vec == int))
}
```

b. Then, generate a random vector of 100 integers (in a range 1-20) use the function to count the number of times the number 12 is in that vector.

```r
set.seed(123)
randomVec <- sample(1:20, 100, replace = TRUE)
target <- 12
times <- no_of_int(randomVec, target)
times_fast <- no_of_int_fast(randomVec, target)

table(randomVec)
```

```
## randomVec
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  1  3  6  2  6  4  8  6  5  8  4  5  3 10  7  3  5  4  6  4
```

```r
times
```

```
## [1] 5
```

```r
times_fast
```

```
## [1] 5
```

Part 2 Write a function that takes in a data.frame as an input, prints out the column names, and returns its dimensions.

```r
colname_dim <- function(df){
  print(colnames(df))
  return(dim(df))

}

# mtcars is a built in dataset
test <- colname_dim(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```r
test
```

```
## [1] 32 11
```

Exercise 6: Apply family functions

Part 1

Below we print six first rows of the built-in dataset, mtcars, from the 1974 Motor Trend US magazine, which comprises information on the fuel consumption and 10 aspects of automobile design and performance for 32 selected car models.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Use `apply()` function to find the standard deviation and the 0.8-quantile of each of the automobile characteristic.

```
apply(mtcars, 2, function(x) sd(x))
```

```
##         mpg         cyl        disp          hp        drat          wt
##   6.0269481   1.7859216 123.9386938  68.5628685   0.5346787   0.9784574
##        qsec          vs          am        gear        carb
##   1.7869432   0.5040161   0.4989909   0.7378041   1.6152000
```

```
apply(mtcars, 2, function(x){ quantile(x, 0.8)})
```

```
##     mpg     cyl    disp      hp    drat      wt    qsec      vs      am    gear
##  24.080   8.000 350.800 200.000   4.048   3.770  19.332   1.000   1.000   4.000
##    carb
##   4.000
```

Part 2

Below is a vector of dates in year 2018.

```
set.seed(1234)
y2018 <- seq(as.Date("2018-01-01", format = "%Y-%m-%d"),
             as.Date("2018-12-31", format = "%Y-%m-%d"),
             "days")
length(y2018)
```

```
## [1] 365
```

```
# A random sample of 10 dates from 2018
y2018_sample <- sample(y2018, size = 10)
y2018_sample
```

```
##  [1] "2018-10-11" "2018-12-02" "2018-04-11" "2018-04-21" "2018-05-13"
##  [6] "2018-04-08" "2018-04-13" "2018-08-02" "2018-03-31" "2018-11-22"
```

Use an apply family function to return the number of weeks left from each day in y2018_sample to the New Year, 2019/01/01.

```
ny2019 <- as.Date("2019-01-01", format = "%Y-%m-%d")
weeks_to_2019 <- sapply(y2018_sample, function(x) ceiling((ny2019-x) / 7))
names(weeks_to_2019) <- y2018_sample
weeks_to_2019
```

```
## 2018-10-11 2018-12-02 2018-04-11 2018-04-21 2018-05-13 2018-04-08 2018-04-13
##         12          5         38         37         34         39         38
## 2018-08-02 2018-03-31 2018-11-22
##         22         40          6
```