HW 2 More on R

STAT 5400

Due: Sep 13, 2024 9:30 AM

Submit your solutions as an .Rmd file and accompanying .pdf file. Include all the **relevant** R code and output. With the `echo` option in R markdown, you opt to hide some R code that is not very relevant (but still run it to generate the document).

Always comment on your result whenever it is necessary. For example, in problem 3, display and also comment on the Q-Q plot that you produce

Reading assignments.

Read Chapters 2-4, 7 of *Using R for Data Analysis and Graphics.* https://cran.r-project.org/doc/contrib/usingR.pdf.

Problems

1. Use the `system.time` function in R to time the performance of the same task in two different ways:

- Generate a vector of 500,000 random variates from a Normal (0, 1) density and use the `sum` function to calculate their sum.

```r
system.time(sum(rnorm(500000, mean = 0, sd = 1)))
```

```
##    user  system elapsed
##    0.05    0.00    0.05
```

- Create a variable called `answer` and initialize it to 0. Then, using a `for` loop, do the following steps 500,000 times: generate a single Normal (0, 1) value and add it to sum contained in `answer`.

```r
system.time({
  answer = 0
  for (i in 1:500000) answer = answer + rnorm(1, mean = 0, sd = 1)
})
```

```
##    user  system elapsed
##    0.68    0.39    1.20
```

- Besides including the R code and output, compare on the relevant timings for both methods and state which one is more efficient.

Calling the function once to generate a rnorm series is more efficient rather than making multiple function calls

2. Use R to do the following:

- Create a matrix called M with the following entries:

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}.$$

```
M <- matrix(c(1:9), nrow = 3, ncol = 3)
```

- Create a vector called v with the following entries: `17 46 181`.

```
v = c(17,46,181)
```

- Compute and display the product Mv produced by matrix multiplication.

```
Mv = M %*% v
print(Mv)
```

```
##      [,1]
## [1,] 1468
## [2,] 1712
## [3,] 1956
```

- Compute and display the transpose of M.

```
print(t(M))
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

- Display only those elements of v that have values less than 50.

```
print(v[v > 50])
```

```
## [1] 181
```

3.

- Use either `help.search` function or just google it to locate a package that contains a function to compute the skewness of a vector of numbers. Make sure that it uses the standard definition of skewness. What is the name of the function, and which package is it in?

```r
moments::skewness(x, na.rm = TRUE)
# Calculate the skewness of a vector
```

- Locate an R function that computes the five-number summary of a vector of numbers. What is the name of the function, and which package is it in?

```r
stats::fivenum()
# Calculate the min, 25th precentile, median, 75th  percentile and max values
```

- Write an R function that does the following:
  - Accepts one argument: a vector.
  - Checks whether the vector is numeric.
  - If not, displays the message `Vector must be numeric` and exist.
  - If yes, computes the skewness of the values (after removing any missing values).
    * If the absolute value of skewness is less than 1, returns a list containing two objects: skewness in an object named `skewness`; a vector consisting of the mean and standard devviation in an object named `dsescstats`.
    * Otherwise, returns a list containing two objects: skewness in an object named `skewness`; a vector consisting of the five-number summary in an object named `descstats`.

```r
myfunc <- function(vec) {
  if(!is.numeric(vec)){
    print("`Vector must be numeric` and exist. ")
    return()
  }
  skewness = moments::skewness(vec, na.rm = TRUE)

  if(abs(skewness) < 1) {
    return(list(skewness = skewness, dsescstats = c(mean(vec), sd(vec))))
  } else {
        return(list(skewness = skewness, dsescstats = stats::fivenum(vec)
))

  }

}
```

```
+ Run your function in R three times, using the following vectors as arguments:
    + `c("stat", "actuarial", "2022")`
    + `rnorm(100)`
    + `rexp(5)`
+ In the document that you submit for homework, show both the R code and output for the three calls to
```

```r
myfunc(c("stat", "actuarial", "2022"))
```

```
## [1] "`Vector must be numeric` and exist. "
```

```
## NULL
```

```r
myfunc(rnorm(100))
```

```
## $skewness
## [1] 0.2240835
##
## $dsescstats
## [1] -0.1282338  0.9680089
```

```r
myfunc(rexp(5))
```

```
## $skewness
## [1] 0.12435
##
## $dsescstats
## [1] 0.8883546 0.7250281
```

4. Review the example on two-sample t-test on slide 45 of S2P1.pdf. If you are not familiar with t-test, you may google some online tutorials.

- Imagine we have run 1000 experiments (rows) and each of which collects data on 50 individuals (columns). The first 25 individuals in each experiment are assigned to group 1 and the rest to group 2.
- You may imagine in practice we only have 50 observations, 25 of which belongs to group 1. With simulations, we have luxury to generate 1000 samples, each of which has 50 observations, to explore the distribution of t-statistics.
  - We first generate some random data to represent this problem.

```r
set.seed(1)
m <- 1000
n <- 50
X <- matrix(rnorm(m * n, mean=10, sd=3), nrow=m)
grp <- rep(1:2, each=n/2)
```

- For the first sample (i.e., the first row), compute the t-statistic manually. Compare your result with the output from the following code.

```r
t.test(X[1, grp==1], X[1, grp==2])$stat
```

```
##          t
## -0.5284632
```

- Use R to compute the t-statistics for all 1000 samples. Try to optimize your code to make it efficient.
- Check the R code in the end of section 24.7 on https://adv-r.hadley.nz/perf-improve.html#t-test. Use `system.time` to compare the computing time of the R function provided on the webpage with your own function. Comment on why your code is slower or faster. Don't check their code before you finish writing your own code.

```r
# Custom two sample t-test
customTwoT<- function(a,b) {
  if(!is.numeric(a) && !is.numeric(b)) {print( 'one is not numeric')
    return()
  }
  return(
    (mean(a)- mean(b))/
      sqrt( ((var(a))/length(a)) + ((var(b))/length(b))))
}


#First Row
customTwoT(X[1,1:25], X[26,50])
```

```
## [1] NA
```