STAT 385 Fall 2019 - Homework Assignment 03

Tianli Ding

Due by 12:00 PM 10/12/2019

Grader Comment

Efficiency:

The code and document is written with lots of logical thought about the tasks with few unnecessary or unnecessarily verbose lines and is written in a least brute force way.

Correctness:

The files run with no errors and the coding is written in a most accurate way.

Documentation:

The explanations and comments contain reasonable clarity and exhibit mid-tier (to be expected) fundamentals of usefulness, grammar, smoothness, and structure.

Beauty:

The document is most visually appealing and has the strongest readability. The coding output and visualizations are very attractive.

##

Please note that the summary above applies to the completed work. If the submission was incomplete, the ultimate mark for the submission will be scaled down proportionally.

Please take care to ensure that one's repo includes the image files that are included in one's .Rmd file. Otherwise, the graded file that is returned to the student may not contain the images.

You may find that your returned work (.Rmd and .pdf) has had sections removed. This is only for accelerating the grading process by removing code chunks that are impeding knitting. If you believe that you were incorrectly penalized for missing work, please do contact the TA and/or professor.

Please see comments below for specifics.

The Homework Problems

Below you will find problems for you to complete as an individual. It is fine to discuss the homework problems with classmates, but cheating is prohibited and will be harshly penalized if detected.

1. Create a custom volume measurement function that will convert the following units of volume:

a. 13 imperial (liquid) cups to cubic inches.
b. 2.5 US customary (liquid) gallons to fluid ounces.
c. 3 US customary (dry) teaspoons to milliliters.
d. 75 (dry) liters to imperial quarts.

```r
converter <- function(type1, type2, x){
a = "cups"
b = "cubic inches"
c = "gallons"
d = "ounces"
e = "teaspoons"
f = "milliliters"
g = "liters"
h = "quarts"
if(type1 == a && type2 == b) return(x*14.4375)
else if(type1 == c && type2 == d) return(x*128)
else if(type1 == e && type2 == f) return(x*4.92892)
else if(type1 == g && type2 == h) return(x*0.879877)
else return("wrong type")
}
## a.
converter("cups", "cubic inches", 13)
```

```
## [1] 187.6875
```

```r
## b.
converter("gallons", "ounces", 2.5)
```

```
## [1] 320
```

```r
## c.
converter("teaspoons", "milliliters", 3)
```

```
## [1] 14.78676
```

```r
## d.
converter("liters", "quarts", 75)
```

```
## [1] 65.99077
```

2. Do the following:

a. create a 25×25 matrix with autoregressive structure with $p = 9/10$, every element in the matrix should be equal to $(9/10)^{|i-j|}$ where `i` is the row index and `j` is the column index. Report the row and column sums of this matrix.

```
A = matrix(rep(0,625), 25,25)
for(i in 1:25){
  for(j in 1:25){
    A[i,j] = (9/10)^(abs(i-j))
  }
}
#column sum and row sum
apply(A,2,sum)
```

```
##  [1]  9.282102 10.102336 10.823706 11.454229 12.000910 12.469823 12.866179
##  [8] 13.194382 13.458077 13.660195 13.802983 13.888025 13.916268 13.888025
## [15] 13.802983 13.660195 13.458077 13.194382 12.866179 12.469823 12.000910
## [22] 11.454229 10.823706 10.102336  9.282102
```

```
apply(A,1,sum)
```

```
##  [1]  9.282102 10.102336 10.823706 11.454229 12.000910 12.469823 12.866179
##  [8] 13.194382 13.458077 13.660195 13.802983 13.888025 13.916268 13.888025
## [15] 13.802983 13.660195 13.458077 13.194382 12.866179 12.469823 12.000910
## [22] 11.454229 10.823706 10.102336  9.282102
```

b. run the commands:

```
set.seed(13)
x <- c(10, 10)
n <- 2
```

Create a while loop which concatenates a new mean-zero normal random variables that have $\sigma = 2$ to the existing vector x at every iteration. Have this loop terminate when the standard error (estimated standard deviation of x divided by $\sqrt{n}$) is lower than $1/10$. Report n.

```
while(1){
  x = append(x, rnorm(1,0,2))
  n = n+1
  se = sd(x)/sqrt(n)
  if(se < 1/10) break;
}
print(n)
```

```
## [1] 483
```

c. repeat part **b** and report n after running the commands:

```
set.seed(13)
x <- rnorm(1, 0, 2)
n <- 1
```

```
while(1){
  x = append(x, rnorm(1,0,2))
  n = n+1
  se2 = sd(x)/sqrt(n)
```

```r
  if(se2 < 1/10) break;
}
print(n)
```

```
## [1] 436
```

d. The sample size required to get a standard error lower than 1/10 was smaller in part **c** than it was in part **b**. We would expect for this to be the case before we ran any code. Why?

Since the **b** starts with two 10s. Then we add the third value (one mean-zero normal random variable which have sigma = 2) which will lead to large sd, thus indicating we will have large se as well. However, we start with one (mean-zero normal random variable which have sigma = 2) value, then we add a second value. It won't display large sd and se. Therefore, it takes more values to balance back the sd in question b.

3. Do the following (Efron's bootstrap):

a. load in the dataset dataHW3.csv

```r
dataset = read.csv("https://uofi.box.com/shared/static/mwntzgp2rvyewf292k6i62pykjz1onnw.csv")
```

b. call the first column of this dataset x. Compute the statistic `(mean(x) - 10)/se(x)` where se is shorthand for standard error (see the previous problem for the definition of standard error).

```r
x = dataset[,1]
se <- function(x){ return(sd(x)/sqrt(length(x)))}
stat = (mean(x)-10)/se(x)
print(stat)
```

```
## [1] -0.1915867
```