STAT 210
Applied Statistics and Data Analysis: Homework 1
Solution

Due on Sept. 14/2025

Question 1

(a) Create a vector named `v1` with a sample of size 200 from the set {`one, two, three, four, five`} (use the words, not the numbers). The values `one` and `five` should have probability 0.16, values `two` and `four` probability 0.2, and value `three` probability 0.28.

(b) Create an **ordered** factor named `fact1` using the vector `v1` as input. The levels should be in increasing order, in the sense that `one` is less than `two`, and so on.

(c) Change the labels for the categories. Use the names `awful, poor, normal, good`, and `excellent` for the categories `one`, `two`, `three`, `four`, and `five`, respectively. One way to do this is to use the `labels` augument in the `factor` function to rename the labels. Look up the help page for `factor`. Name the resulting ordered factor `fact2`.

(d) Create a vector named `v2` with a sample of size 200 from the set {`yellow`, `green`, `blue`, `red`}. All the values should have the same probability.

(e) Create a factor named `fact3` with vector `v2` as input

(f) Create a data frame named `df1` with two components. The first component should be named `item1` and should have the content of `fact1`. The second component should be named `item2` and should have the content of `fact3`.

(g) Use the function `table` to create a table for the two factors in `df1`, i.e., you should get a table of `item1` against `item2`.

Solution

(a) We use the function `sample` to create `v1`.

```
v1 <- sample(c('one','two','three','four','five'),200,TRUE,c(0.16,0.2,0.28,0.2,0.16))
str(v1)
```

```
##  chr [1:200] "three" "four" "three" "three" "one" "four" "five" "five" ...
```

Observe that the components of the vector have mode character.

(b) For the ordered factor we use the function `ordered`.

```
fact1 <- ordered(v1,levels = c('one','two','three','four','five'))
str(fact1)
```

```
##  Ord.factor w/ 5 levels "one"<"two"<"three"<..: 3 4 3 3 1 4 5 5 2 4 ...
```

(c) Use `labels` as suggested:

```r
fact2 <- factor(v1,levels = c('one','two','three','four','five'),
                labels = c('awful','poor','normal','good',
                           'excellent'))
str(fact2)
```

```
##  Factor w/ 5 levels "awful","poor",..: 3 4 3 3 1 4 5 5 2 4 ...
```

(d) Create vector v2:

```r
v2 <- sample(c('yellow','green','blue','red'), 200, TRUE)
str(v2)
```

```
##  chr [1:200] "green" "green" "green" "green" "red" "green" "yellow" ...
```

(e) Create factor `fact3`:

```r
fact3 <- factor(v2)
str(fact3)
```

```
##  Factor w/ 4 levels "blue","green",..: 2 2 2 2 3 2 4 4 3 4 ...
```

(f) Create data frame `df1`:

```r
df1 <- data.frame(item1 = fact2, item2 = fact3)
str(df1)
```

```
## 'data.frame':    200 obs. of  2 variables:
##  $ item1: Factor w/ 5 levels "awful","poor",..: 3 4 3 3 1 4 5 5 2 4 ...
##  $ item2: Factor w/ 4 levels "blue","green",..: 2 2 2 2 3 2 4 4 3 4 ...
```

(f) Finally, we use `table`.

```r
table(df1)
```

```
##            item2
## item1       blue green red yellow
##   awful       10    10   6      7
##   poor        16     6   8     11
##   normal      16    12  11     15
##   good         9     9  15      9
##   excellent    2     9  10      9
```

Question 2

In this question we want use the MonteCarlo method to estimate the value of e. The exercise is based on the article *Estimating the Value of e by Simulation* by G.K. Russel, published in The American Statistician in February, 1991. This paper is available from the BB page for the course. Russel's paper is based in turn on an exercise in a book by B.V. Gnedenko.

Gnedenko's exercise asks the reader to show that if $U_1, U_2, \dots$ are iid uniformly distributed on $(0,1)$, $S_n = \sum_1^n U_i$, and N is the smallest value of n for which $S_n > 1$, then $E(N) = e$. We will assume this result to be true. You may try proving this but this is not part of your homework. I give a few hints below, but you can find a proof in Russel's paper. We will use this result to get a MonteCarlo approximation for e, similar to what we did in class for π.

(a) We start by building a series of commands to obtain the value of N for a simulated sample using the control function `while`. Before the control function, initialize a counter `N` at zero, that will keep track of the number of variables we add until the sum is above 1, and a variable `S`, also at zero, that will store the sum of the uniform random variables. As argument of the command `while` (i.e., within parenthesis) write the condition that the sum is less than or equal to 1. While this condition is satisfied, the code to be run (within braces) should add a uniform random variable to `S` and update the index `N`. After the braces, print `N`.

(b) The result of (a) is a single simulation of N. We need to do this a large number of times and then calculate the average value to use the MC method to approximate e. Write a `for` loop that will repeat the simulation in (a) `k` times. The loop should store the `k` simulated values for `N` in a vector.

(c) Use the result of (b) to simulate (`k` =) 10,000, 100,000, and one million times the value of `N`. Calculate the approximate value of e for the three values of `k`, and also the error in the approximation. Produce a table of relative frequencies for the values of `N` you stored and plot a bar diagram for the relative frequency table for `N`. Comment on your results.

Hints for the proof of $E(N) = e$. Observe that for any integer n, $N = n$ if and only if $S_n > 1$ but $S_{n-1} \leq 1$. Show that $P(N = n) = P(S_{n-1} < 1) - P(S_n < 1)$. So we need to calculate $P(S_n < 1)$, and this is harder. It can be shown that $P(S_n < 1) = 1/n!$. Use this to obtain $P(N = n)$, and once you have the probability function for N, calculate its expected value.

Solution

(a) The code is

```
N <- 0
S <- 0
while(S <= 1){
  S <- S + runif(1)
  N <- N+1
}
print(N)
```

If we run the code we get, for instance,

```
## [1] 2
```

The result is random, so different runs may give different results.

(b) The code is

```
res <- numeric(k)
for (i in 1:k) {
  N <- 0
  S <- 0
  while(S <= 1){
    S <- S + runif(1)
    N <- N+1
  }
res[i] <- N
}
```

The result is a vector `res` of dimension `k`, with simulated values of `N`. The variable `k` controls the size of the simulation.

(c) We now use the previous code to approximate the value of e. Remember that $e \approx 2.71828$. The results are for 10,000, 100,000 and one million, respectively, and in each case the value of the error is calculated.

```
k <- 10000
res <- numeric(k)
for (i in 1:k) {
  N <- 0
  S <- 0
```

```r
  while(S <= 1){
    S <- S + runif(1)
    N <- N+1
  }
res[i] <- N
}
mean(res)
```

```
## [1] 2.735
```

```r
mean(res) - exp(1)
```

```
## [1] 0.01671817
```

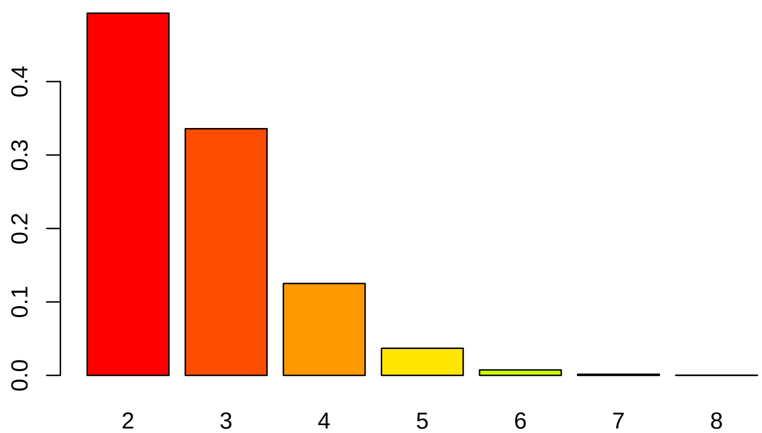
The relative frequency table is

```r
table(res)/k
```

```
## res
##      2      3      4      5      6      7      8
## 0.4931 0.3358 0.1251 0.0369 0.0074 0.0015 0.0002
```

and the barplot is

```r
barplot(table(res)/k, col = rainbow(20))
```



Similarly, for `k = 100000`,

```r
k <- 100000
```

```
## [1] 2.72008
```

```
## [1] 0.001798172
```

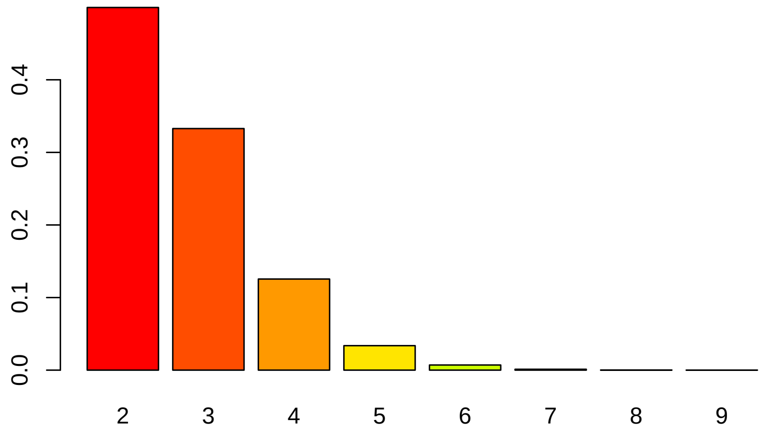
The frequency table is

```r
table(res)/k
```

```
## res
##       2       3       4       5       6       7       8       9
## 0.49960 0.33279 0.12550 0.03370 0.00705 0.00119 0.00015 0.00002
```

and the barplot is

```r
barplot(table(res)/k, col = rainbow(20))
```



And for k = one million

```
k <- 1000000
```

```
## [1] 2.717125
```

```
## [1] -0.001156828
```

The frequency table is

```
table(res)/k
```

```
## res
##        2        3        4        5        6        7        8        9
## 0.500594 0.333228 0.124550 0.033332 0.006896 0.001211 0.000162 0.000026
##       10
## 0.000001
```

and the barplot is

```
barplot(table(res)/k, col = rainbow(20))
```