STAT 210
Applied Statistics and Data Analysis:
Homework 1

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.

```r
words <- c("one", "two", "three", "four", "five")
probabilities <- c(0.16, 0.20, 0.28, 0.20, 0.16)

v1 <- sample(words, 200, replace = TRUE, prob = probabilities)
head(v1)
```

```
## [1] "three" "four"  "one"   "three" "five"  "three"
```

(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.

```r
fact1 <- factor(v1, levels = c("one", "two", "three", "four", "five"), ordered = TRUE)
head(fact1)
```

```
## [1] three four  one   three five  three
## Levels: one < two < three < four < five
```

(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`.

```r
fact2 <- factor(v1, levels = c("one", "two", "three", "four", "five"), labels = c("awful", "poor", "nor
head(fact2)
```

```
## [1] normal    good      awful     normal    excellent normal
## Levels: awful < poor < normal < good < excellent
```

(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.

```r
colors <- c("yellow", "green", "blue", "red")
v2 <- sample(colors, 200, replace = TRUE)
head(v2)
```

```
## [1] "yellow" "red"    "green"  "green"  "green"  "blue"
```

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

```r
fact3 <- factor(v2, levels = c("yellow", "green", "blue", "red"))
head(fact3)
```

```
## [1] yellow red    green  green  green  blue
## Levels: yellow green blue red
```

(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`.

```r
df1 <- data.frame(item1 = fact1, item2 = fact3)
head(df1)
```

```
##   item1  item2
## 1 three yellow
## 2  four    red
## 3   one  green
## 4 three  green
## 5  five  green
## 6 three   blue
```

(f) 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`.

```r
table(df1$item1, df1$item2)
```

```
##
##         yellow green blue red
##   one        8    11    3   7
##   two        8     5   10  17
##   three     13    13   13  17
##   four      14    10    7  13
##   five      11    11    5   4
```

Question 2

In this question we want use simulation (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 $U_1, U_2, \ldots$ 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`.

```r
N <- 0
S <- 0

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

```
## [1] 3
```

(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.

```r
k <- 1e6
N_list <- numeric(k)

for(i in 1:k){
  N <- 0
  S <- 0

  while (S <= 1) {
    S <- S + runif(1)
    N <- N + 1
  }
  N_list[i] <- N
}
head(N_list)
```

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

(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.

```
mean(N_list)
```

```
## [1] 2.719299
```

```
error <- abs(mean(N_list) - exp(1))

error
```

```
## [1] 0.001017172
```

```
rel_table <- table(N_list) / length(N_list)
rel_table
```

```
## N_list
##        2        3        4        5        6        7        8        9
## 0.499510 0.333202 0.125703 0.033291 0.006881 0.001212 0.000176 0.000022
##       10
## 0.000003
```

```
barplot(rel_table, main = "Relative Frequency of N", ylab = "Proportion")
```