MAS115 R programming

Lab Class 4
Homework Solutions

1 Cumulative sums of random variables

(a) **Pseudo-code**

SET $s \leftarrow 0, n \leftarrow 0$
WHILE ($s \leq 100$)
    SAMPLE x from Exponential(1/5)
    $s \leftarrow x + s$
    $n \leftarrow n + 1$
ENDWHILE
RETURN n

(b) Translating into R code:

```
# While loop to count sum of exponential random variables
sum <- 0
count <- 0
while(sum <= 100) {
  sum <- sum + rexp(1, rate = 0.2)
  count <- count + 1
}
count
```

It would probably be best to solve the second part of the question by writing the above as a function which could be called as we wished (see Lab Class 5). However if we wanted to embed the code in a `for` loop then we can write:

```
# Putting it inside a for loop
n <- rep(NA, 1000)
for(i in 1:1000) {
  sum <- 0
  count <- 0
  while(sum <= 100) {
    sum <- sum + rexp(1, rate = 0.2)
    count <- count + 1
  }
  n[i] <- count
}
hist(n)
```

Better style would be something like this.

```
Niter <- 1000
target <- 100
expmean <- 5
# Main for loop
n <- rep(NA, Niter)
for(i in 1:Niter) {
  # Generate one count
  sum <- 0
  count <- 0
  while(sum <= target) {
    sum <- sum + rexp(1, rate = 1/expmean)
    count <- count + 1
  }
  n[i] <- count
}
hist(n)
```

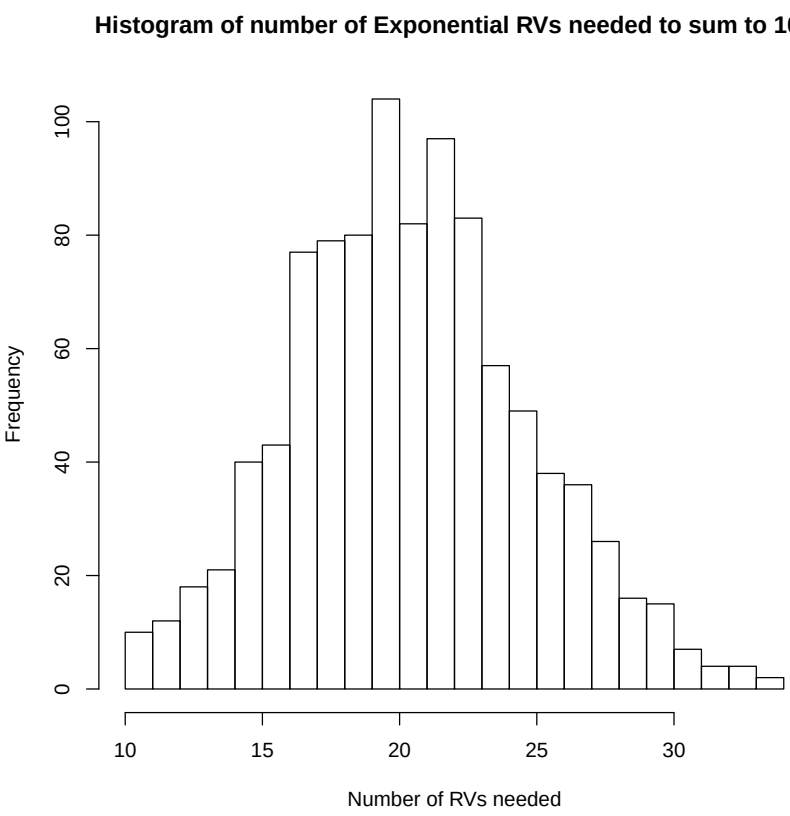

Figure 1: Distribution of numbers of exponential random variables.

Then `head(n)` gives

```
[1] 20 22 21 22 25 21
```

(c) We can estimate the probability that $n > 30$ using `mean(n>30)` which gave 0.016. Based on a much bigger run, the true value is close to 0.022.

2 Death at a chemical plant

(a) As in Lab Class 3, there are two possible approaches here.

Using a `for` loop With a `for` loop our code might look like:

```
## Using a for loop ##
y <- 100
z <- 40
N <- 10000 # Number of realisation to create
C <- rep(NA, N) # Vector to store created values
Q <- 100
h <- 50
for(i in 1:N) {
 # Sample values for u, sy and sz
  u <- rlnorm(1,2,0.1^0.5)
  sy <- rlnorm(1,10,0.2^0.5)
  sz <- rlnorm(1,5,0.05^0.5)
 # Store in C[i] value of conc. at site of interest
  C[i] <- Q/(2*pi*u*(sz*sy)^0.5)*exp(-0.5*(y^2/sy+(z-h)^2/sz))
}
# Plot histogram
hist(C)
# Find mean
mean(C)
# Find 95th quantile
quantile(C, 0.95)
```

Here we create 10,000 values for C individually by cycling through the `for` loop a lot of times.

Using vectorisation We can however create 10,000 values for C simultaneously by taking advantage of vectorisation as follows. You should again run both versions of the code and see how much faster the vectorised version runs (you could try a larger value for N if it's not totally clear).

```
## Using vectorisation
y <- 100
z <- 40
N <- 10000 # Number of realisation to create
Q <- 100
h <- 50
# Note: no need to initialise C
# Create values for u, sy and sz
u <- rlnorm(N,2,0.1^0.5)
sy <- rlnorm(N,10,0.2^0.5)
sz <- rlnorm(N,5,0.05^0.5)
    # Create a C for each
C <- Q/(2*pi*u*(sz*sy)^0.5)*exp(-0.5*(y^2/sy+(z-h)^2/sz))
# Plot histogram
hist(C)
# Find mean
mean(C)
# Find 95th quantile
quantile(C,0.95)
```

Either way of coding performs the same algorithm and, if you chose N very large, the same solution. Some sample output:

```
> head(C)
[1] 0.0005725069 0.0007296576 0.0010734219 0.0006582339 0.0004505959 0.0007169896
```

A histogram of my values for C is shown in Figure 2

(b) The mean is around 0.000694, based on a larger sample and using `mean(C)`.

(c) The mean is around 0.00114, based on a larger sample and using `quantile(C, 0.95)`.

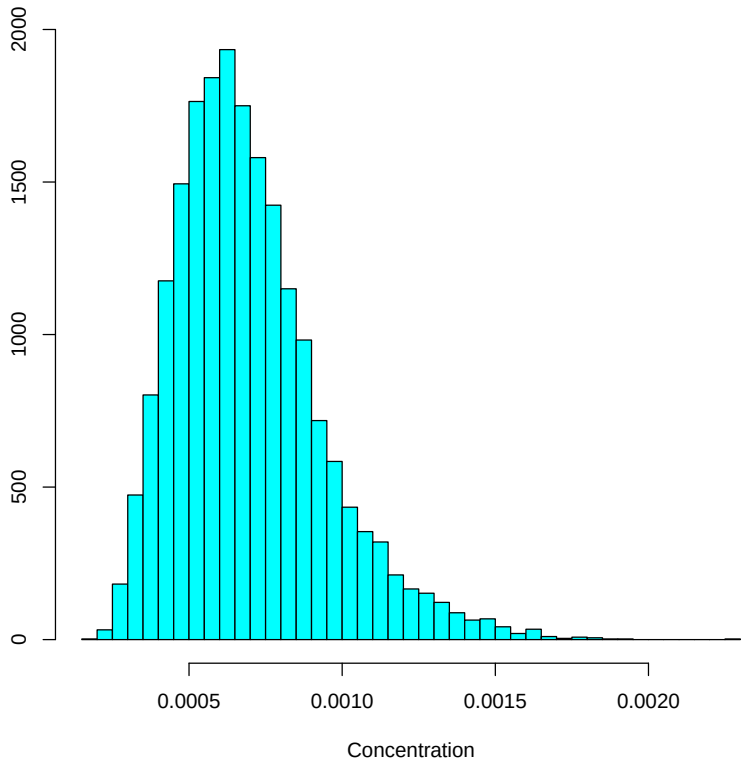

Figure 2: Histogram of the simulated values of the concentration $C(100, 40)$ based on 10,000 random samples.

3 Triangular numbers

To "record" the values mentioned in the question, we could print them out, or store them in an R object, or both.

The solution here does both, using a matrix with named columns to store the values. Storing them as vectors would be fine too.

```
target <- 7
k <- 0
T_k <- 0 # kth triangular number
countSquares <- 0 # count of cases for T_k = perfect square
sols <- matrix(NA,nrow=target,ncol=3,
               dimnames=list(NULL,c("k","T_k","sqroot")))
while(countSquares<target){
  k <- k + 1
  T_k <- T_k + k # T_k is sum of integers 1,2,...,k
  sqroot <- T_k^0.5
  if(sqroot == round(sqroot)){ # T_k is a perfect square
    cat("k = ",k,"  T_k = ",T_k," = ",sqroot,"squared\n")
    countSquares <- countSquares + 1
    sols[countSquares,] <- c(k,T_k,sqroot)
  }
}
```

Output:

```
k =  1   T_k =  1  =  1 squared
k =  8   T_k =  36  =  6 squared
k =  49   T_k =  1225  =  35 squared
k =  288   T_k =  41616  =  204 squared
k =  1681   T_k =  1413721  =  1189 squared
k =  9800   T_k =  48024900  =  6930 squared
k =  57121   T_k =  1631432881  =  40391 squared
```

```
> sols
         k        T_k sqroot
[1,]     1          1      1
[2,]     8         36      6
[3,]    49       1225     35
[4,]   288      41616    204
[5,]  1681    1413721   1189
[6,]  9800   48024900   6930
[7,] 57121 1631432881  40391
```