MAS115 R programming, Homework Solutions 3

Bryony Moody (Original solutions by Prof. P. G. Blackwell)

2020-21

1 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)
  }
}
```

```
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
```

2 Cumulative Sums

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

Translating into R code

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

```
[1] 17
```

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 6). However if we wanted to embed the code in a `for` loop then we can write:

```
# Putting it inside a for loop
set.seed(1)
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
}
head(n)
```

```
[1] 17 25 21 12 20 27
```

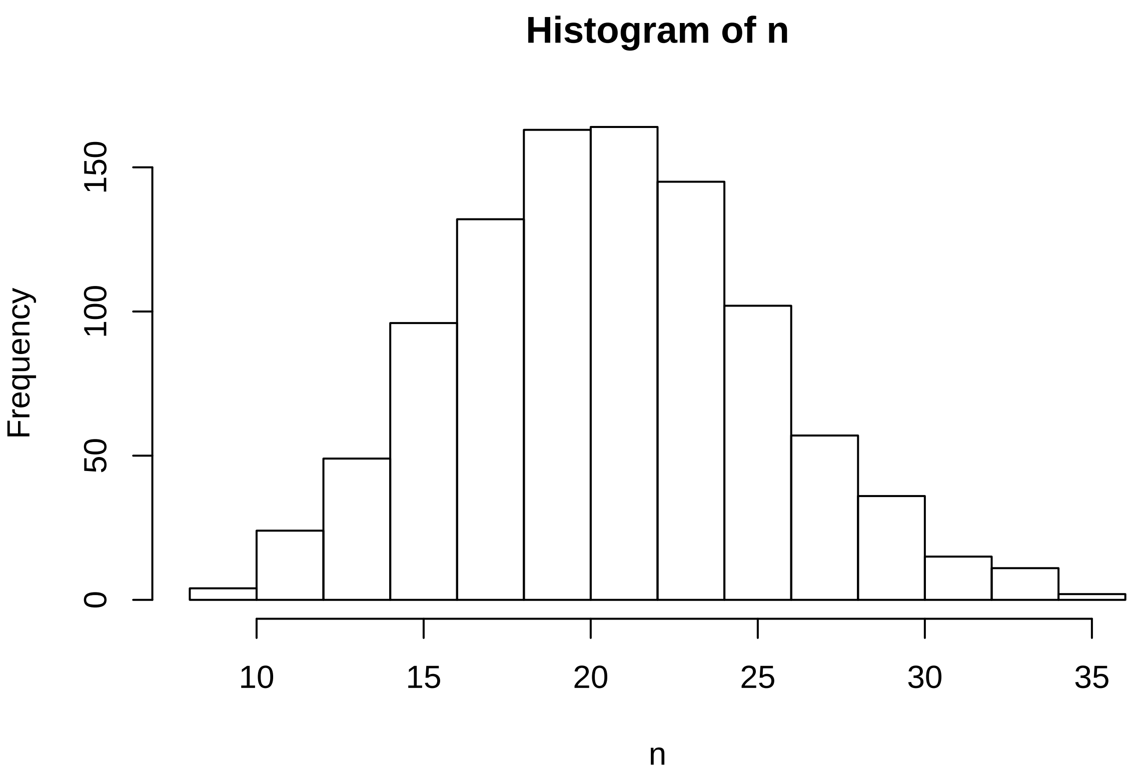
We can estimate the probability that $n > 30$ as follows.

```
mean(n>30)
```

```
[1] 0.028
```

Based on a much bigger run, the true value is close to 0.022. Make sure that you understand exactly how the previous command works; there is quite a bit going on in that one line.

```
hist(n)
```



Better style would be something like this.

```
set.seed(1)
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
}
head(n)
```

```
[1] 17 25 21 12 20 27
```

3 Monte Carlo Integration

The notation here largely follows that in the lecture. If we are really only interested in the interval 0 to 1, we could omit a and b and the corresponding arguments to `runif`, but writing it like this makes it easy to experiment with other cases. On the other hand, the form of the function f is hard-coded in this version, though it is easy to change the exponent. More general code would write the mathematical f as an R function.

Experiment with some other cases!

```
set.seed(0)
n <- 10^6
a <- 0
b <- 1
power <- 2
x <- runif(n,min=a,max=b)
fx <- x^power
Integral <- sum(fx)*(b-a)/n
print(Integral)
```

```
[1] 0.3332849
```