# Suggested answer to exercise 8, chapter 8

## from Ross S.: "Simulation" - 5th edition

Author: Nicklas S. Andersen

Date: April 27, 2016

---

**Exercise 8.a** In this exercise we want to estimate the constant $e^1 = e \approx 2.71828$, by generating $U_1, U_2, ..., U_n$ iid uniform random variables on $[0, 1]$. We define an estimator and random variable

$$N = \min\left\{ n : \sum_{i=1}^{n} U_i > 1 \right\}$$

By running $p = 1000$ simulations we obtain $N_1, N_2, ..., N_p$ iid random variables, where the simulation estimate of $e$ is given by

$$\hat{e} = E\left[\bar{N}\right] = \frac{1}{p}\sum_{i=1}^{p} N_i$$

To obtain an actual estimate of the constant $e$, we use the following function:

```
#-------------------------------------------------------------
# Function: estimate.e(maxSim)
#           - (Function to estimate the constant 'e')
# Input   : maxSim
#           - (The nr. of r.v N to generate)
# Output  : est
#           - (The estimate of the constant 'e')
#-------------------------------------------------------------
estimate.e <- function(maxSim){
  storeVal <- c()
  for(i in 1:maxSim){                #Number of simulations.
    U1 <- 0
    n <- 0
    while(U1 < 1){                   #Sum values of uniform r.v numbers
                                     #until the sum exceeds 1, then return n.
      U1 <- U1 + runif(1)
      n <- n + 1
    }
    storeVal <- c(n, storeVal)
    est <- sum(storeVal)/maxSim      #Calculate expectation.
  }
  return(est)
}
estimate.e(100000)
```

```
## [1] 2.71628
```

---

**Exercise 8.b** By slightly changing the above function we are also able to obtain an independent estimate of the variance and give a 95% confidence interval estimate of $e$ . We do this by using the function:

```
#-------------------------------------------------------------
# Function: estimate.e(maxSim)
#           - (Function to estimate the constant 'e')
# Input   : maxSim
#           - (The nr. of r.v N to generate)
# Output  : est
#           - (The estimate of the constant 'e'
#              and sample variance)
#-------------------------------------------------------------
estimate.e <- function(maxSim){
  storeVal1 <- c()
  storeVal2 <- c()
  for(j in 1:2){             #Run simulations twice to estimate expected value
                             #and sample variance independently.
    for(i in 1:maxSim){      #Number of simulations.
      U1 <- 0
      n <- 0
      while(U1 < 1){         #Sum values of uniform r.v numbers
                             #until the sum exceeds 1, then return n.
        U1 <- U1 + runif(1)
        n <- n + 1
      }
      if(j == 1){            #Collect values of r.v n  (to calculate expected value).
        storeVal1 <- c(n, storeVal1)
      }
      else{                  #Collect values of r.v n  (to calculate sample variance).
        storeVal2 <- c(n, storeVal2)
      }
    }
    if(j == 1){              #Calculate expected value
      estExpVal <- sum(storeVal1)/maxSim
    }
    else {                   #Calculate sample variance
      estVar <- sum((storeVal2 - estExpVal)^2)/(maxSim - 1)
    }
  }
  result <- c(estExpVal, estVar, maxSim)
  return(result)
}
tmp <- estimate.e(100000)
print(paste(c("Estimate: ", round(tmp[1], digits = 8)), collapse = " "))
```

```
## [1] "Estimate:  2.71518"
```

```
print(paste(c("Sample variance: ", round(tmp[2], digits = 8)), collapse = " "))
```

```
## [1] "Sample variance:  0.76307277"
```

```r
#--------------------------------------
# 95% confidence interval estimate of e
# (traditional normal-based)
#--------------------------------------
print(paste(c("Lower: ", round(tmp[1] - 1.96*sqrt(tmp[2])/sqrt(tmp[3]),
                               digits = 8)), collapse = " "))
```

```
## [1] "Lower:  2.70976574"
```

```r
print(paste(c("Upper: ", round(tmp[1] + 1.96*sqrt(tmp[2])/sqrt(tmp[3]),
                               digits = 8)), collapse = " "))
```

```
## [1] "Upper:  2.72059426"
```