Assignment 2: Loops

- **By Sandeep Joshi**

Question 1: Loops

```
#-----------------------------------#
#     A S S I G N M E N T   2       #
#-----------------------------------#

# Set working directory
cat("\014")
setwd("D:/Stevens/Sem 3/FE515/Week 2")

# Read the csv files
dow <- read.csv("DOW.csv")
sp500 <- read.csv("SP500.csv")
# check the column names
names(dow)
names(sp500)
# get the length of the second vector to avoid calc inside loop
sp500_size <- length(sp500$Ticker.symbol)
# mandatory for loop
linebreak = ''
answer = ''
for (ticker in dow$Ticker)
{
    for (index in 1:sp500_size)
    {
        if (ticker == sp500$Ticker.symbol[index])
        {
            print(paste(ticker, index, sep = '--'))

            # Another way of doing things but curiously enough it suppresses
            # linefeed character if it's at the end. Any place else works
            # which is rather inconvenient
            #if (length(answer) > 0)
            #{
            #    linebreak = "\n"
            #}
            #answer <- cat(answer, linebreak, ticker, "--", index)
            break # to quickly exit as soon we find first match
        }
    }
}

# Without using loop
print("Without using loop")
matches <- match(dow$Ticker, sp500$Ticker.symbol)
cat(paste("\n", dow$Ticker, '--', matches))
```

Console D:/Stevens/Sem 3

```
[1] "AXP--29"
[1] "BA--70"
[1] "CAT--88"
[1] "CSCO--104"
[1] "CVX--98"
[1] "DD--152"
[1] "DIS--474"
[1] "GE--203"
[1] "GS--210"
[1] "HD--225"
[1] "IBM--238"
[1] "INTC--236"
[1] "JNJ--249"
[1] "JPM--252"
[1] "KO--112"
[1] "MCD--293"
[1] "MMM--1"
[1] "MRK--299"
[1] "MSFT--303"
[1] "NKE--326"
[1] "PFE--359"
[1] "PG--374"
[1] "T--46"
[1] "TRV--440"
[1] "UNH--455"
[1] "UTX--458"
[1] "V--469"
[1] "VZ--466"
[1] "WMT--472"
[1] "XOM--176"
```

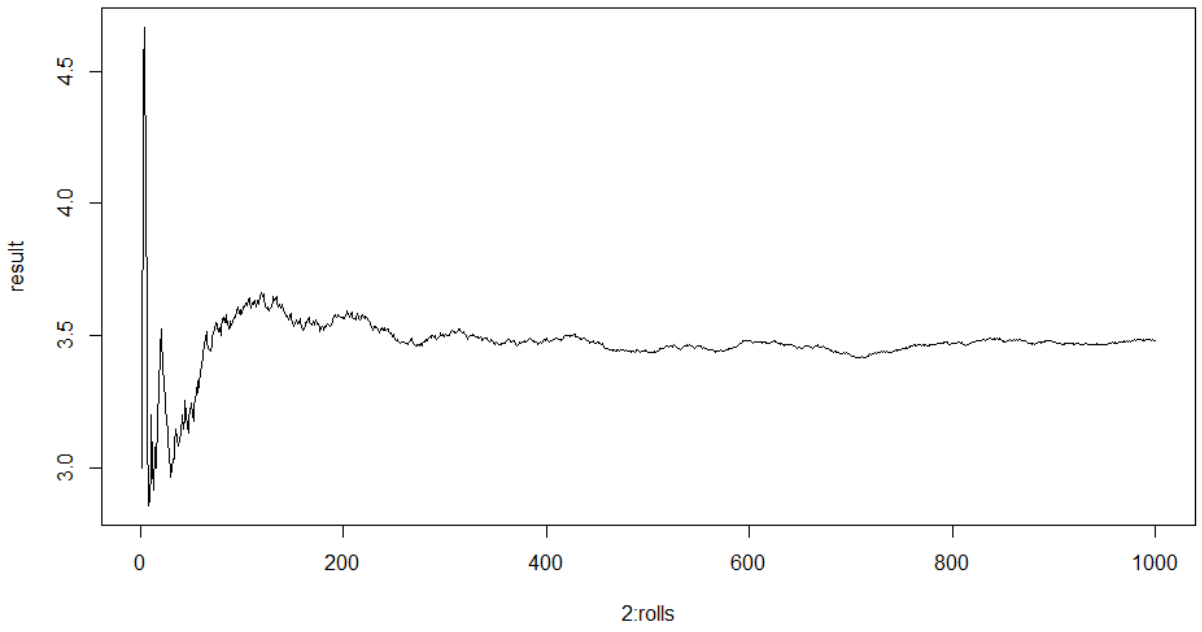
```
> # Without using loop
> print("Without using loop")
[1] "Without using loop"
> matches <- match(dow$Ticker, sp500$Ticker.symbol)
> cat(paste("\n", dow$Ticker, '--', matches))

 AXP -- 29
 BA -- 70
 CAT -- 88
 CSCO -- 104
 CVX -- 98
 DD -- 152
 DIS -- 474
 GE -- 203
 GS -- 210
 HD -- 225
 IBM -- 238
 INTC -- 236
 JNJ -- 249
 JPM -- 252
 KO -- 112
 MCD -- 293
 MMM -- 1
 MRK -- 299
 MSFT -- 303
 NKE -- 326
 PFE -- 359
 PG -- 374
 T -- 46
 TRV -- 440
 UNH -- 455
 UTX -- 458
 V -- 469
 VZ -- 466
 WMT -- 472
 XOM -- 176
```

Question 2: User Defined Functions

Fair Dice

```
118 # Fair Dice
119 fair_Dice <- function(rolls)
120 {
121     result <- NULL
122     dice_total <- 0
123     for (roll in 1:rolls)
124     {
125         tmp <- sample(x=c(1,2,3,4,5,6), size = 1, replace = T, prob = rep(1/6, 6))
126         if (roll > 1)
127         {
128             result <- c(result, dice_total/ (roll - 1))
129         }
130         dice_total <- dice_total + tmp  # this is done after we calculate the n-1 mean as asked in
131     }
132
133    plot(2:rolls, result, type = "l")
134    cat("For ", rolls, " rolls, series converged at :", result[length(result)])
135 }
136 fair_Dice(1000)
```

```
> fair_Dice(1000)
For  1000  rolls, series converged at : 3.482482
```



P.S. According to my interpretation of the problem statement we are asked to find mean of all n-1 rolls at nth roll. Hence, dice_total is computed after vector has been saved.

Loaded Dice:

```r
# Loaded Dice
loaded_Dice <- function(rolls, prob)
{
    result <- NULL
    dice_total <- 0
    # Compute the last probablity if only 5 or less
    diff = 6 - length(prob)
    if (diff > 0)
    {
        prob <- append(prob, rep(((1 - sum(prob))/diff), diff)) # distributing remanant probablities
    }

    for (roll in 1:rolls)
    {
        tmp <- sample(x=c(1,2,3,4,5,6), size = 1, replace = T, prob = prob)
        dice_total <- dice_total + tmp  # this is done after we calculate the n-1 mean
    }

    return(dice_total/rolls)
}

# Only four probablities are given in the function call
cat("Mean of all rolls is:", loaded_Dice(1000, c(1/5, 1/5, 1/4, 1/8)))
```

```
> # Only four probablities are given in the function call
> cat("Mean of all rolls is:", loaded_Dice(1000, c(1/5, 1/5, 1/4, 1/8)))
Mean of all rolls is: 3.076
```

The remaining probabilities are evenly distributed. This dice was loaded on the lesser numbers hence as expected the mean is less than 3.5.

Question 3: Expectation value for dice rolls

Strategy:

Calculate the expectation value for a dice roll:

E(v) = $\sum_{k=1}^{6} \left(P_{\ k}\right)k$ now P_k for all values of dice is 1/6, so

= 1/6 *(1 + 2 + 3 + 4 + 5 + 6) = 21/6 = 3.5

So, this means at every roll our expectation value would be 3.5 but this is for one independent roll.

When we get multiple rolls we could increase our chance for expectation value by making judicious decision and for that we need to find a pivot point.

Let's consider value 3.5 which is mid-range as well as expectation value. Since we could only have 3 or 4 (real numbers) as mid points. Let's start with 4.

If we get 4 should we go for a re-roll? And the answer is yes as we have half the chance for getting either from the set [1,2,3] or from set [4,5,6]. Which means that there's .5 probability of ruining our score but there's .5 also that it would increase or stay same.

Let's calculate E(3) and E(4) for choosing pivot at 3 and 4.

E(3):

i) First roll got less than 3: P(<3) = 2/6 = 1/3
   a. Reroll expected value = 3.5
ii) First roll got greater than or equal to 3: P(>=3) = 4/6 = 2/3

(3 + 4 + 5 + 6)*1/4 = 4.5

Combined expected value = 1/3(3.5) + 2/3(4.5) = 1.1667 + 3 = 4.1667

E(4):

i) First roll got less than 4: P(<4) = 3/6 = 1/2
   a. Reroll expected value = 3.5
ii) First roll got greater than or equal to 4: P(>=4) = 3/6 = 1/2

(4 + 5 + 6)*1/3 = 5

Combined expected value = 1/2(3.5) + 1/2(5) = 8.5/2 = 4.25

E(5):

i) First roll got less than 5: P(<5) = 4/6 = 2/3
   a. Reroll expected value = 3.5
ii) First roll got greater than or equal to 5: P(>=5) = 2/6 = 1/3

(5 + 6)*1/2 = 5.5

Combined expected value = 2/3(3.5) + 1/3(5.5) = 12.5/3 = 4.1667

Similarly, we can see for E(2) = 3.916, E(6) =3.916. So we can see that our intuition was correct. Similar results were obtained from running the simulations for different pivot points.

```
# Highest payoff

# We know that max expectation value for a dice roll is 3.5, so we will
# employ using decision pivot at value 4 i.e if first roll is less than 4
# we will roll again. Explanation is in the Report

fair_Dice <- function(rolls, pivot)
{
    result <- NULL
    dice_total <- 0
    for (roll in 1:rolls)
    {
        tmp <- sample(x=c(1,2,3,4,5,6), size = 1, replace = T, prob = rep(1/6, 6))
        if (tmp < pivot)
        {
            # re-roll
            tmp <- sample(x=c(1,2,3,4,5,6), size = 1, replace = T, prob = rep(1/6, 6))
        }
        dice_total <- dice_total + tmp
    }
    return(dice_total/rolls)

}
rolls = 10000
cat("For ", rolls, " rolls, series average is: ", fair_Dice(rolls, 4))
```

Output

For 4:

```
> rolls = 10000
> cat("For ", rolls, " rolls, series average is: ", fair_Dice(rolls, 4))
For  10000  rolls, series average is:  4.2511
```

For 5:

```
> rolls = 10000
> cat("For ", rolls, " rolls, series average is: ", fair_Dice(rolls, 5))
For  10000  rolls, series average is:  4.1727
```