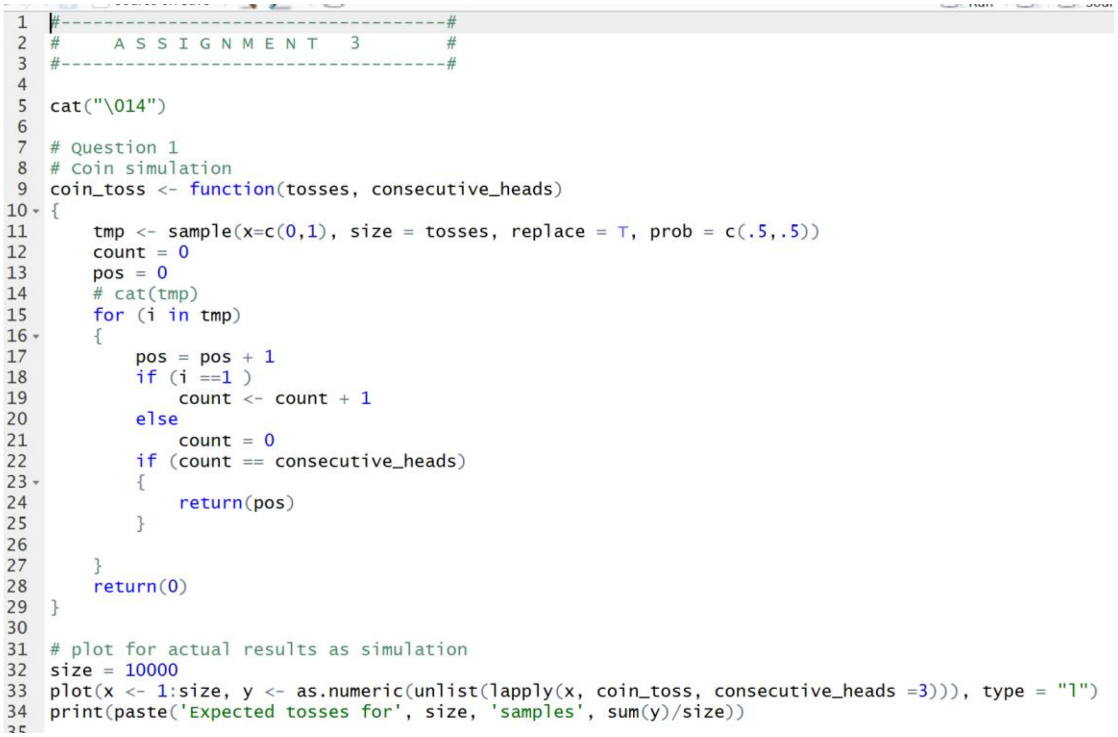
Assignment 3: Loops

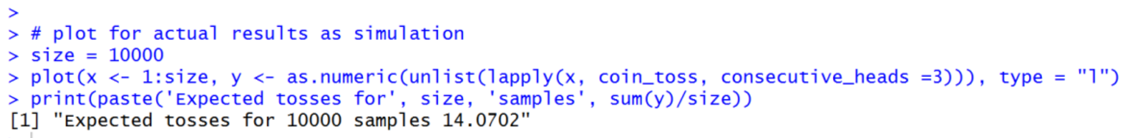
- **By Sandeep Joshi**

Question 1: Coin simulation

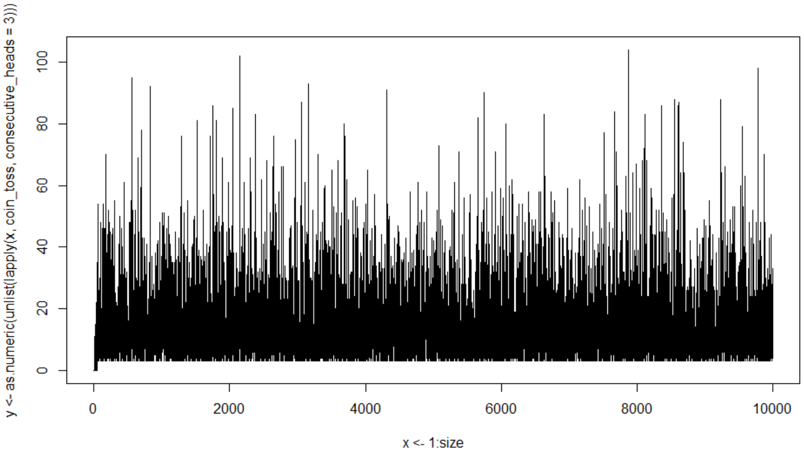


```
#-------------------------------------#
#     A S S I G N M E N T   3         #
#-------------------------------------#

cat("\014")

# Question 1
# Coin simulation
coin_toss <- function(tosses, consecutive_heads)
{
    tmp <- sample(x=c(0,1), size = tosses, replace = T, prob = c(.5,.5))
    count = 0
    pos = 0
    # cat(tmp)
    for (i in tmp)
    {
        pos = pos + 1
        if (i ==1 )
            count <- count + 1
        else
            count = 0
        if (count == consecutive_heads)
        {
            return(pos)
        }

    }
    return(0)
}

# plot for actual results as simulation
size = 10000
plot(x <- 1:size, y <- as.numeric(unlist(lapply(x, coin_toss, consecutive_heads =3))), type = "l")
print(paste('Expected tosses for', size, 'samples', sum(y)/size))
```

Output:



```
>
> # plot for actual results as simulation
> size = 10000
> plot(x <- 1:size, y <- as.numeric(unlist(lapply(x, coin_toss, consecutive_heads =3))), type = "l")
> print(paste('Expected tosses for', size, 'samples', sum(y)/size))
[1] "Expected tosses for 10000 samples 14.0702"
```



Bonus Question

Let the expectation be x

For three consecutive heads we need at least three tosses.

Toss 1 if tail. We need at least one more toss hence, multiply that with prob(T)

=> $(x + 1) * (1/2)$ -----(1)

Toss 2 if tail i.e. head, tail, we need two more tosses, multiply that with prob(H)*prob(T)

=> $(x + 2) * (1/2) * (1/2)$ -----(2)

Toss 3 if tail i.e head, head, tail, three more tosses, and prob(H)*prob(H)*prob(T)

=> $(x + 3)*(1/2)*(1/2)*(1/2)$ -----(3)

If all heads i.e. head, head, head. no more tosses and prob prob(H)^3

=> $3*(1/8)$ ----(4)

In all these equations the first part is the number of tosses it would take for 3 heads and the second part is the net probability of it happening.

Adding (1), (2), (3) and (4) we would get x,

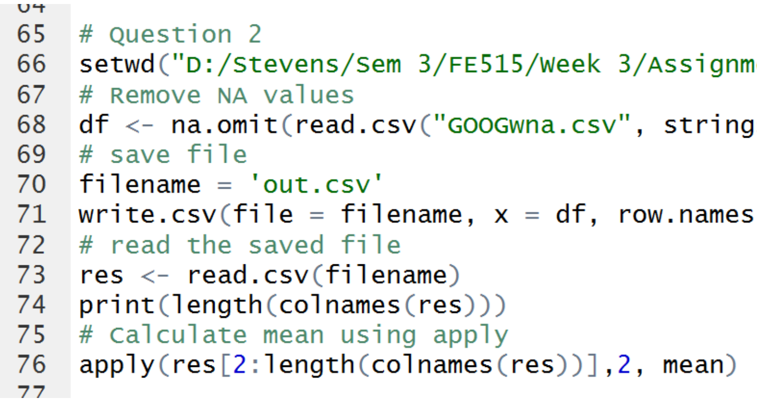
$x - (x + 1)*(1/2) - (x + 2)*(1/4) - (x + 3)*(1/8) - 3/8 = 0$

$x - x*(7/8) - (7/4) = 0$

$x = 7*8/4 = 14$

Our answer from 10000 samples was 14.072 which is approx. 14 as shown above.

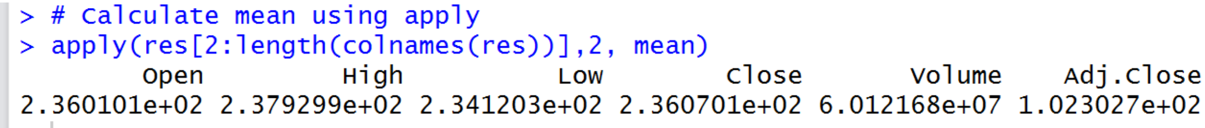
Question 2: Removing NA values



```r
# Question 2
setwd("D:/Stevens/Sem 3/FE515/Week 3/Assignment 3")  # Comment this line
# Remove NA values
df <- na.omit(read.csv("GOOGwna.csv", stringsAsFactors = F))
# save file
filename = 'out.csv'
write.csv(file = filename, x = df, row.names = FALSE)
# read the saved file
res <- read.csv(filename)
print(length(colnames(res)))
# Calculate mean using apply
apply(res[2:length(colnames(res))],2, mean)
```

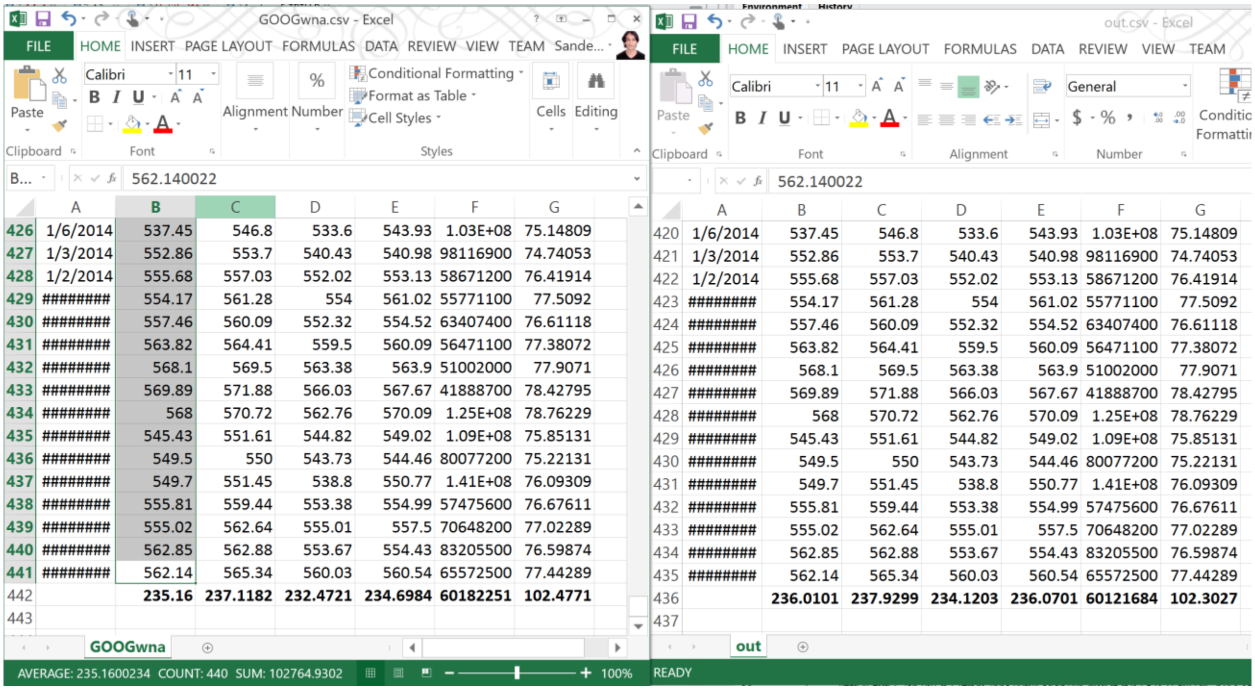
P.S. Since I was not provided with the file, I created one based on the files given in other assignment and manually changes some values in it to NA. Attached is the screenshot from the file. Both the input and output files are part of the submission as well.

Output:



```
> # Calculate mean using apply
> apply(res[2:length(colnames(res))],2, mean)
        Open         High          Low        Close       Volume    Adj.Close 
2.360101e+02 2.379299e+02 2.341203e+02 2.360701e+02 6.012168e+07 1.023027e+02 
```

On the left is the input file and on the right is the output file. The means are shown in the bottom and number of lines suppressed could be checked.



B... 562.140022

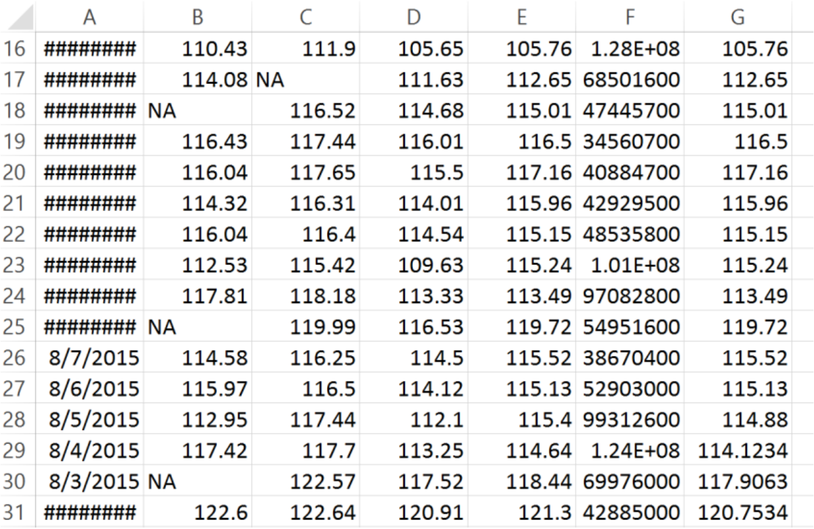
	A	B	C	D	E	F	G
426	1/6/2014	537.45	546.8	533.6	543.93	1.03E+08	75.14809
427	1/3/2014	552.86	553.7	540.43	540.98	98116900	74.74053
428	1/2/2014	555.68	557.03	552.02	553.13	58671200	76.41914
429	########	554.17	561.28	554	561.02	55771100	77.5092
430	########	557.46	560.09	552.32	554.52	63407400	76.61118
431	########	563.82	564.41	559.5	560.09	56471100	77.38072
432	########	568.1	569.5	563.38	563.9	51002000	77.9071
433	########	569.89	571.88	566.03	567.67	41888700	78.42795
434	########	568	570.72	562.76	570.09	1.25E+08	78.76229
435	########	545.43	551.61	544.82	549.02	1.09E+08	75.85131
436	########	549.5	550	543.73	544.46	80077200	75.22131
437	########	549.7	551.45	538.8	550.77	1.41E+08	76.09309
438	########	555.81	559.44	553.38	554.99	57475600	76.67611
439	########	555.02	562.64	555.01	557.5	70648200	77.02289
440	########	562.85	562.88	553.67	554.43	83205500	76.59874
441	########	562.14	565.34	560.03	560.54	65572500	77.44289
442		**235.16**	**237.1182**	**232.4721**	**234.6984**	**60182251**	**102.4771**
443							

AVERAGE: 235.1600234 COUNT: 440 SUM: 102764.9302 100%

562.140022

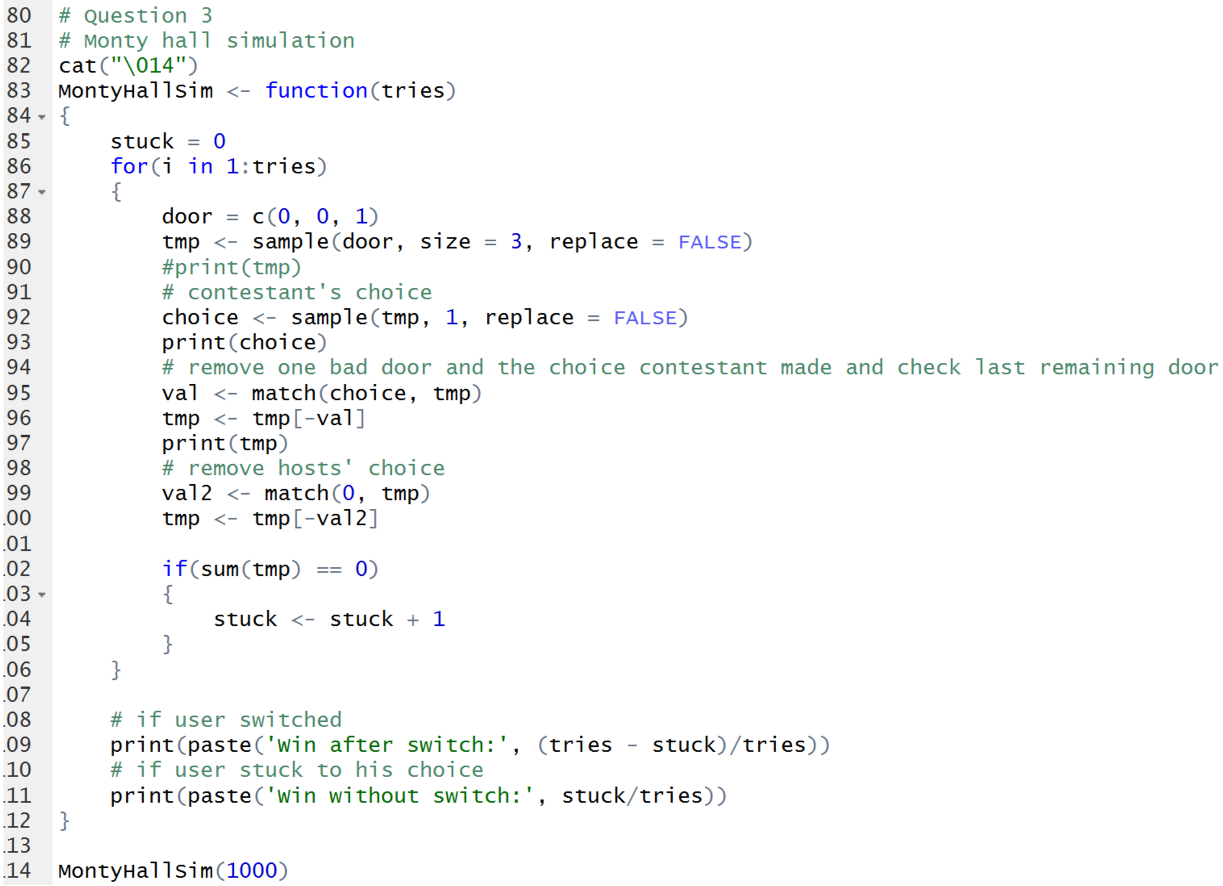
	A	B	C	D	E	F	G
420	1/6/2014	537.45	546.8	533.6	543.93	1.03E+08	75.14809
421	1/3/2014	552.86	553.7	540.43	540.98	98116900	74.74053
422	1/2/2014	555.68	557.03	552.02	553.13	58671200	76.41914
423	########	554.17	561.28	554	561.02	55771100	77.5092
424	########	557.46	560.09	552.32	554.52	63407400	76.61118
425	########	563.82	564.41	559.5	560.09	56471100	77.38072
426	########	568.1	569.5	563.38	563.9	51002000	77.9071
427	########	569.89	571.88	566.03	567.67	41888700	78.42795
428	########	568	570.72	562.76	570.09	1.25E+08	78.76229
429	########	545.43	551.61	544.82	549.02	1.09E+08	75.85131
430	########	549.5	550	543.73	544.46	80077200	75.22131
431	########	549.7	551.45	538.8	550.77	1.41E+08	76.09309
432	########	555.81	559.44	553.38	554.99	57475600	76.67611
433	########	555.02	562.64	555.01	557.5	70648200	77.02289
434	########	562.85	562.88	553.67	554.43	83205500	76.59874
435	########	562.14	565.34	560.03	560.54	65572500	77.44289
436		**236.0101**	**237.9299**	**234.1203**	**236.0701**	**60121684**	**102.3027**
437							

Input files with NA values



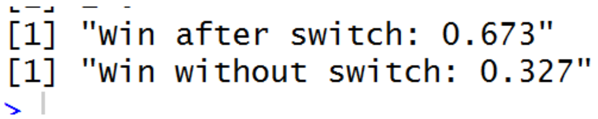
	A	B	C	D	E	F	G
16	########	110.43	111.9	105.65	105.76	1.28E+08	105.76
17	########	114.08	NA	111.63	112.65	68501600	112.65
18	########	NA	116.52	114.68	115.01	47445700	115.01
19	########	116.43	117.44	116.01	116.5	34560700	116.5
20	########	116.04	117.65	115.5	117.16	40884700	117.16
21	########	114.32	116.31	114.01	115.96	42929500	115.96
22	########	116.04	116.4	114.54	115.15	48535800	115.15
23	########	112.53	115.42	109.63	115.24	1.01E+08	115.24
24	########	117.81	118.18	113.33	113.49	97082800	113.49
25	########	NA	119.99	116.53	119.72	54951600	119.72
26	8/7/2015	114.58	116.25	114.5	115.52	38670400	115.52
27	8/6/2015	115.97	116.5	114.12	115.13	52903000	115.13
28	8/5/2015	112.95	117.44	112.1	115.4	99312600	114.88
29	8/4/2015	117.42	117.7	113.25	114.64	1.24E+08	114.1234
30	8/3/2015	NA	122.57	117.52	118.44	69976000	117.9063
31	########	122.6	122.64	120.91	121.3	42885000	120.7534

Question 3: The Monty Hall Simulation



```r
# Question 3
# Monty hall simulation
cat("\014")
MontyHallSim <- function(tries)
{
    stuck = 0
    for(i in 1:tries)
    {
        door = c(0, 0, 1)
        tmp <- sample(door, size = 3, replace = FALSE)
        #print(tmp)
        # contestant's choice
        choice <- sample(tmp, 1, replace = FALSE)
        print(choice)
        # remove one bad door and the choice contestant made and check last remaining door
        val <- match(choice, tmp)
        tmp <- tmp[-val]
        print(tmp)
        # remove hosts' choice
        val2 <- match(0, tmp)
        tmp <- tmp[-val2]

        if(sum(tmp) == 0)
        {
            stuck <- stuck + 1
        }
    }

    # if user switched
    print(paste('Win after switch:', (tries - stuck)/tries))
    # if user stuck to his choice
    print(paste('Win without switch:', stuck/tries))
}

MontyHallSim(1000)
```

Output



```
[1] "Win after switch: 0.673"
[1] "Win without switch: 0.327"
>
```