Final Project

Young Seok Seo

December 15, 2021

```r
#Regret Match algorithms
regret_match <- function(y,Pay) {
  m=dim(Pay)[1]
  n=dim(Pay)[2]
  T = length(y)
  x = matrix(data=NA,nrow=T,ncol=1)
  u = matrix(data=NA,nrow=T,ncol=1)
  q = matrix(data=1,nrow=m,ncol=1)/m
  reg = matrix(data=0,nrow=T,ncol=m)
  q_help = matrix(data=0,nrow=T,ncol=m)


  for (s in 1:T) {
    x[s]=which(rmultinom(1,1,q)==1)
    u[s] = Pay[x[s],y[s]]

    reg[s,]=(Pay[,y[s]]- u[s])/s
    if (s>1) {
      reg[s,]=reg[s,] + reg[s-1,]*((s-1)/s)
    }

    for(col in 1:ncol(reg)) {
      if (reg[s,col] <= 0.0) {
        q_help[s,col] = 0
      }
      else {
        q_help[s,col] = reg[s,col]
      }
    }

    if (max(q_help[s,]) > 0) {
      q = q_help[s,] / (sum(q_help[s,]))
    }
    else {
      q = matrix(data=1,nrow=m,ncol=1)/m
    }


  }
  #Use this return to plot cumulative payout
  return(list(xyu=cbind(x,y,u)))
```

```
  #use this return to plot regret
  #return(list(reg = reg,xyu=cbind(x,y,u)))
}

#Choose dimension one of 3, 5, or 17
d = 3

#function to generate Payout matrix that depends on the dimension.
pay_function <- function(d) {
  PI = diag(d)
  PI<-PI[,(2:dim(PI)[1])]
  help<-dim(PI)[1]-1
  help<-c(1,replicate(help,0))
  PI<-cbind(PI,help)
  colnames(PI) <- NULL
  Pay = diag(d) - 2* inv(diag(d) + PI)
}
Pay <- pay_function(d)

# action sequence for the column player. Transition Matrix A = I
markov_y_identity <- function(d,T) {
  my_num <- 1:d
  trans_mat<-diag(d)
  initial_state <- c(1)
  vector_of_zero <- rep(0,d-1)
  initial_state <- c(initial_state,vector_of_zero)
  initial_state<-sample(initial_state)
  y<-c()
  for (i in 1:T) {
    my_prob <- initial_state %*% (trans_mat %^% i)
    y[i]<-sample(my_num, size = 1,prob = my_prob, replace = TRUE)
  }
  return(y)
}

# action sequence for the column player. Transition Matrix A = 1/d(ee)^T
markov_y_uniform <- function(d,T) {
  my_num <- 1:d
  trans_mat<-matrix(1,d,d)/d
  initial_state <- c(1)
  vector_of_zero <- rep(0,d-1)
  initial_state <- c(initial_state,vector_of_zero)
  initial_state<-sample(initial_state)
  y<-c()
  for (i in 1:T) {
    my_prob <- initial_state %*% (trans_mat %^% i)
    y[i]<-sample(my_num, size = 1,prob = my_prob, replace = TRUE)
  }
  return(y)
}
```

```r
#random action sequence for the column player.
markov_y_random <- function(d,T) {
  my_num <- 1:d
  rand_gen<-runif(d^2)
  dim(rand_gen) <- c(d,d)
  trans<- -log(rand_gen)
  diagonal_mat<-diag(d)*(1/colSums(trans))
  trans_mat<-trans %*% diagonal_mat
  trans_mat<-t(trans_mat)
  initial_state <- c(1)
  vector_of_zero <- rep(0,d-1)
  initial_state <- c(initial_state,vector_of_zero)
  initial_state<-sample(initial_state)
  y<-c()
  for (i in 1:T) {
    my_prob <- initial_state %*% (trans_mat %^% i)
    y[i]<-sample(my_num, size = 1,prob = my_prob, replace = TRUE)
  }
  return(y)
}
```

```r
#Generate the plot for the cumulative payout using identity transition matrix
cum_pay_identity <- function (d) {
  Pay = pay_function(d)
  regret_group <- c()
  realization <- 200
  T <- 1000
  for (i in 1:realization) {
    y<-markov_y_identity(d,T)
    regret_group[i] <- regret_match(y,Pay)
  }

  plot(cumsum(regret_group[[1]][,3]), type = 'l', ylim = c(0,15),xlim = c(0,15), main = "cumulative pay
  for (i in 2:realization) {
    lines(cumsum(regret_group[[i]][,3]), type = 'l')
  }

  cu<-c()
  vec<-c()
  for (i in 1:T) {
    for (j in 1:realization) {
      vec<-c(vec,(cumsum(regret_group[[j]][,3])[i]))
    }
    cu[i]<-mean(vec)
    vec<-c()
  }
  lines(cu,col='red',lwd=4)
}
```

```r
#Generate the plot for the cumulative payout using uniform transition matrix
cum_pay_uniform <- function(d) {
  Pay = pay_function(d)
  regret_group <- c()
```

```
  realization <- 200
  T <- 1000
  for (i in 1:realization) {
    y<-markov_y_uniform(d,T)
    regret_group[i] <- regret_match(y,Pay)
  }

  plot(cumsum(regret_group[[1]][,3]), type = 'l', ylim = c(-80,80), main = "cumulative payout   with uni
  for (i in 2:realization) {
    lines(cumsum(regret_group[[i]][,3]), type = 'l')
  }

  cu<-c()
  vec<-c()
  for (i in 1:T) {
    for (j in 1:realization) {
      vec<-c(vec,(cumsum(regret_group[[j]][,3])[i]))
    }
    cu[i]<-mean(vec)
    vec<-c()
  }
  lines(cu,col='red',lwd=4)
}
```

```
#Generate the plot for the cumulative payout using random transition matrix
cum_pay_random <- function(d) {
  Pay = pay_function(d)
  regret_group <- c()
  realization <- 200
  T <- 1000
  for (i in 1:realization) {
    y<-markov_y_random(d,T)
    regret_group[i] <- regret_match(y,Pay)
  }

  plot(cumsum(regret_group[[1]][,3]), type = 'l', ylim = c(-10,300), main = "cumulative payout with rand
  for (i in 2:realization) {
    lines(cumsum(regret_group[[i]][,3]), type = 'l')
  }

  cu<-c()
  vec<-c()
  for (i in 1:T) {
    for (j in 1:realization) {
      vec<-c(vec,(cumsum(regret_group[[j]][,3])[i]))
    }
    cu[i]<-mean(vec)
    vec<-c()
  }
  lines(cu,col='red',lwd=4)
}
```

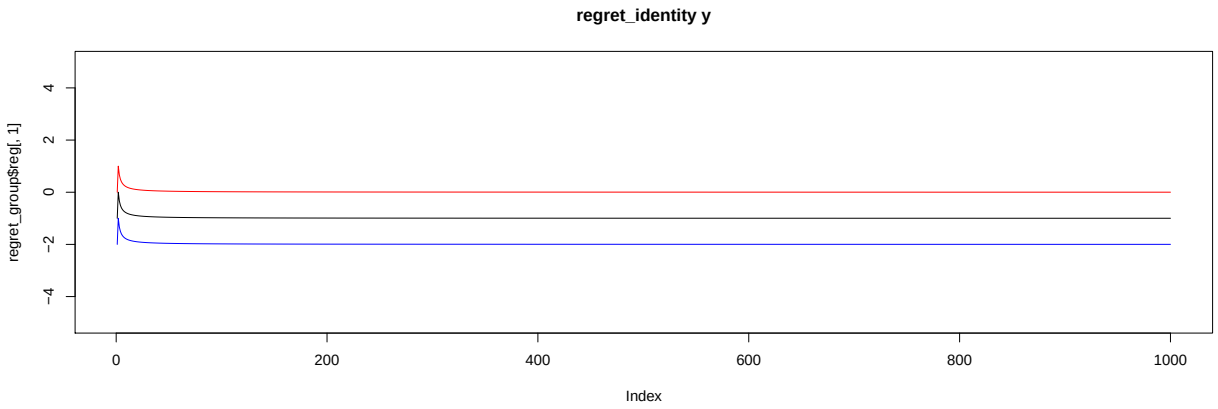
1.What realization length N did you choose for the different d?

I set realization length to 200 and T = 1000, for d = 3, 5 and 17

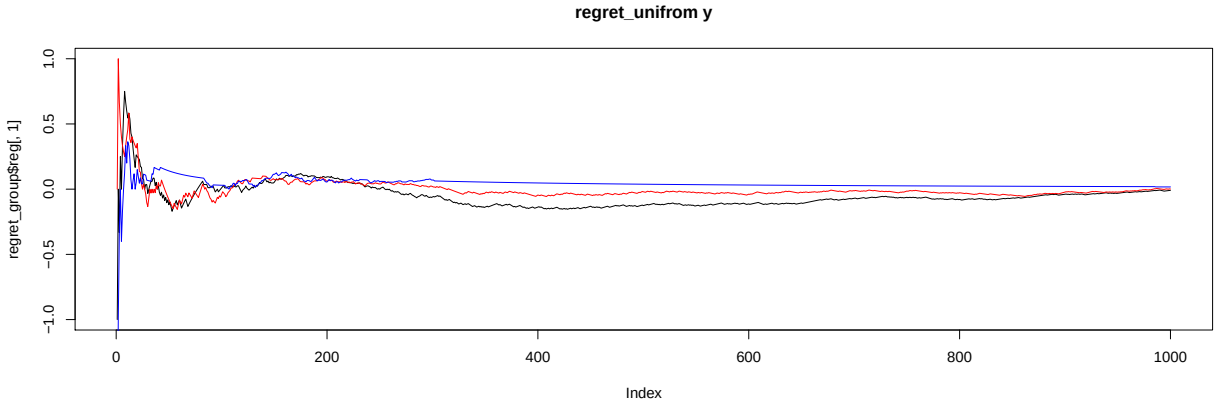
2.How do the pictures you made relate to 'zero average regret'? Hint: you might plot realized regret as a function of time to make your case.

```
#Generate the regret plot
regret_match_reg_plot <- function(y,Pay) {
  m=dim(Pay)[1]
  n=dim(Pay)[2]
  T = length(y)
  x = matrix(data=NA,nrow=T,ncol=1)
  u = matrix(data=NA,nrow=T,ncol=1)
  q = matrix(data=1,nrow=m,ncol=1)/m
  reg = matrix(data=0,nrow=T,ncol=m)
  q_help = matrix(data=0,nrow=T,ncol=m)

  for (s in 1:T) {
    x[s]=which(rmultinom(1,1,q)==1)
    u[s] = Pay[x[s],y[s]]

    reg[s,]=(Pay[,y[s]]- u[s])/s
    if (s>1) {
      reg[s,]=reg[s,] + reg[s-1,]*((s-1)/s)
    }

    for(col in 1:ncol(reg)) {
      if (reg[s,col] <= 0.0) {
        q_help[s,col] = 0
      }
      else {
        q_help[s,col] = reg[s,col]
      }
    }

    if (max(q_help[s,]) > 0) {
      q = q_help[s,] / (sum(q_help[s,]))
    }
    else {
      q = matrix(data=1,nrow=m,ncol=1)/m
    }


  }
  #Use this return to plot cumulative payout
  #return(list(xyu=cbind(x,y,u)))

  #use this return to plot regret
  return(list(reg = reg,xyu=cbind(x,y,u)))
}
T = 1000
```

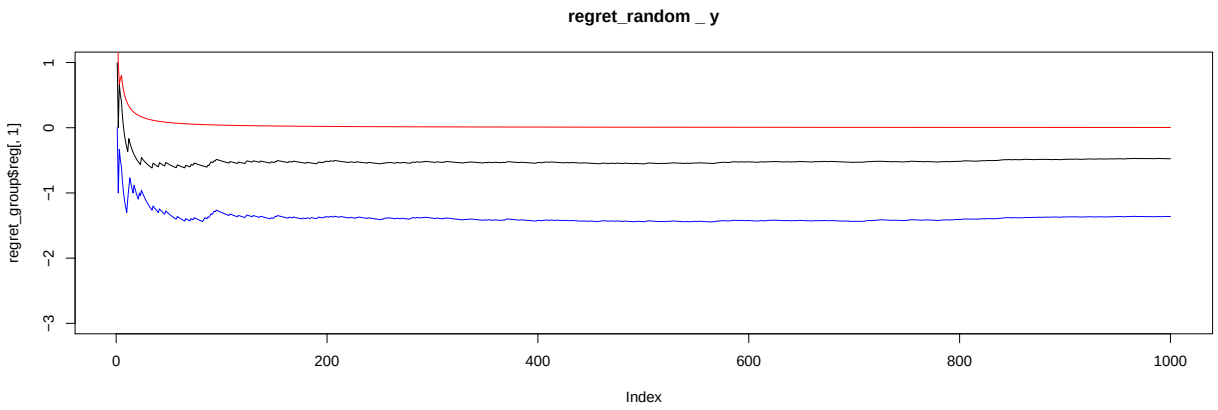
```
y<-markov_y_identity(d,T)
regret_group <- regret_match_reg_plot(y,Pay)
plot(regret_group$reg[,1],type = 'l',ylim = c(-5,5), main = "regret_identity y")
lines(regret_group$reg[,2], col = 'red')
lines(regret_group$reg[,3], col = 'blue')
```




```
T = 1000
y<-markov_y_uniform(d,T)
regret_group <- regret_match_reg_plot(y,Pay)
plot(regret_group$reg[,1],type = 'l',ylim = c(-1,1), main = "regret_unifrom y")
lines(regret_group$reg[,2], col = 'red')
lines(regret_group$reg[,3], col = 'blue')
```




```
T = 1000
y<-markov_y_random(d,T)
Pay = pay_function(d)
regret_group <- regret_match_reg_plot(y,Pay)
plot(regret_group$reg[,1],type = 'l',ylim = c(-3,1), main = "regret_random _ y")
lines(regret_group$reg[,2], col = 'red')
lines(regret_group$reg[,3], col = 'blue')
```



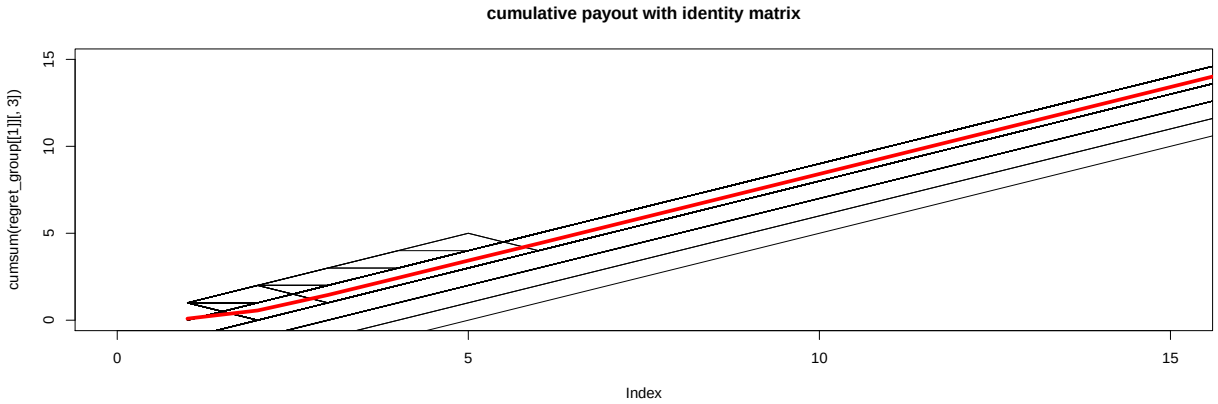
I draw 3 different regret plots. 'regret_identity y' plot is the regret plot when the transition matrix is identity matrix. In that case, the action of the column player is same for all the time-step. So, in my case, if column player plays rock all the time, then row-player does not regret to play 'rock' and 'paper'. However, if row-player plays 'scissor' then it regrets which is the most above line in the plot between 3 different colors.

If transition matrix is A = 1/d(eeˆT), the regret plot shows as 'regret_uniform y' plot, in the beginning of the time step, the plot moves randomly. However, as time step increases, regrets to play is moving together to 0. Their orders are different at each time. The reason that it is keep regretting is it cannot beat column player when the transition matrix is uniform. As you can see the graph of the 'cumulative payout with uniform matrix', we cannot say that row-player wins against column player.

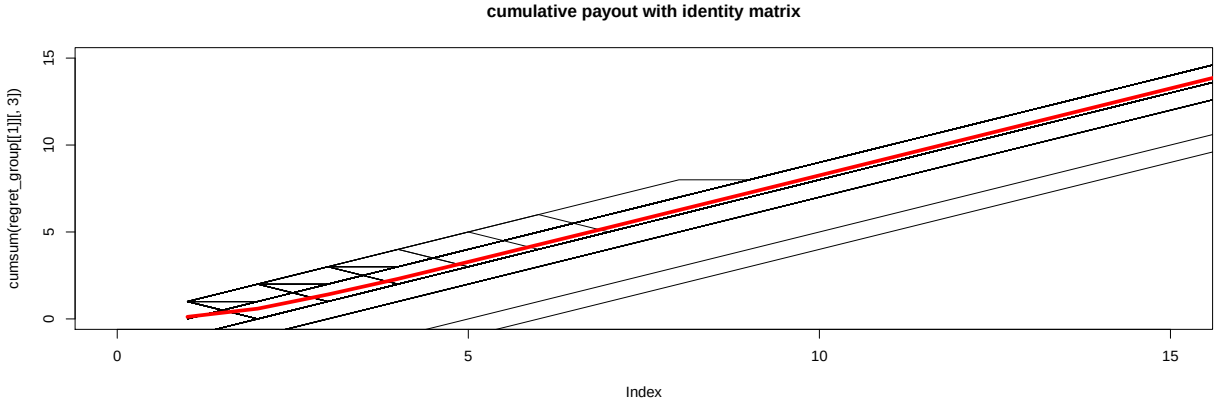
Lastly, row-player does not regret to play two actions as time goes. In the beginning of the time step, the regrets plot can be bouncing a lot. However, after T>400, two actions are below of the 0 in the 'regret_random y', which indicates that row-player does not regret to play two actions. Therefore, row player is winning against column players in the "cumulative payout with random matrix".

3.What is the difference between having many independent sequences of actions from the column player, and only having one?

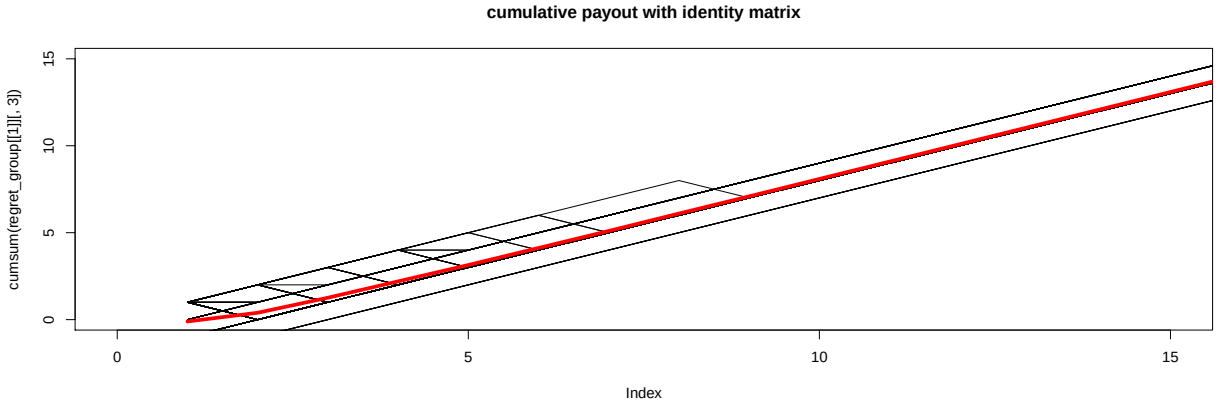
```
cum_pay_identity(3)
```



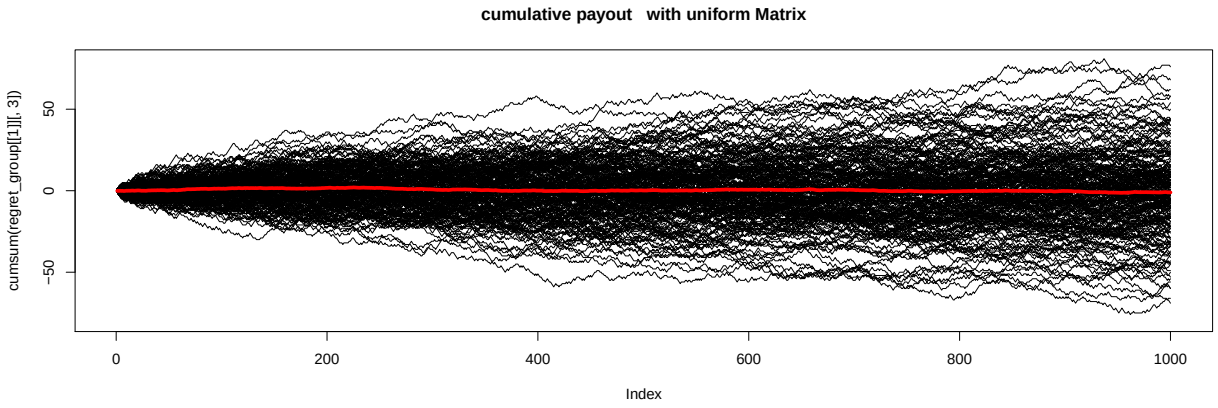

```
cum_pay_identity(5)
```



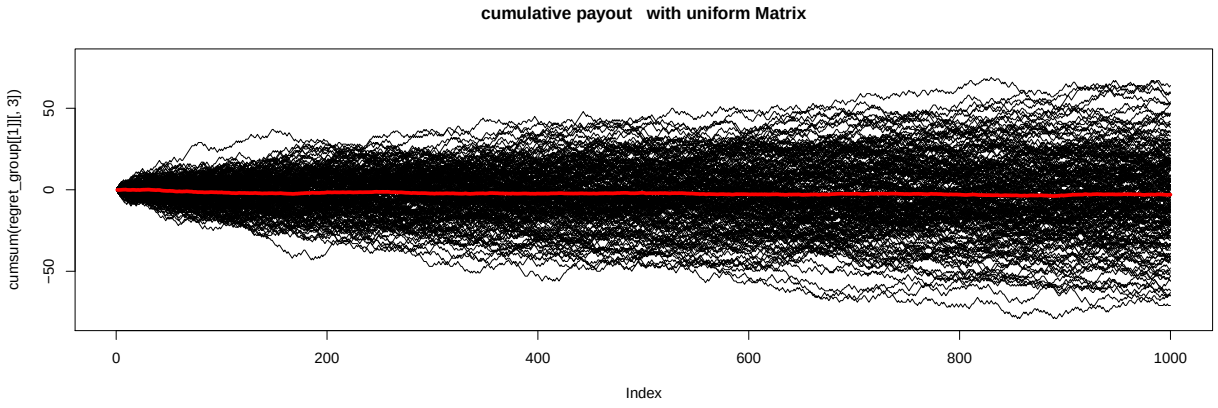

```
cum_pay_identity(17)
```



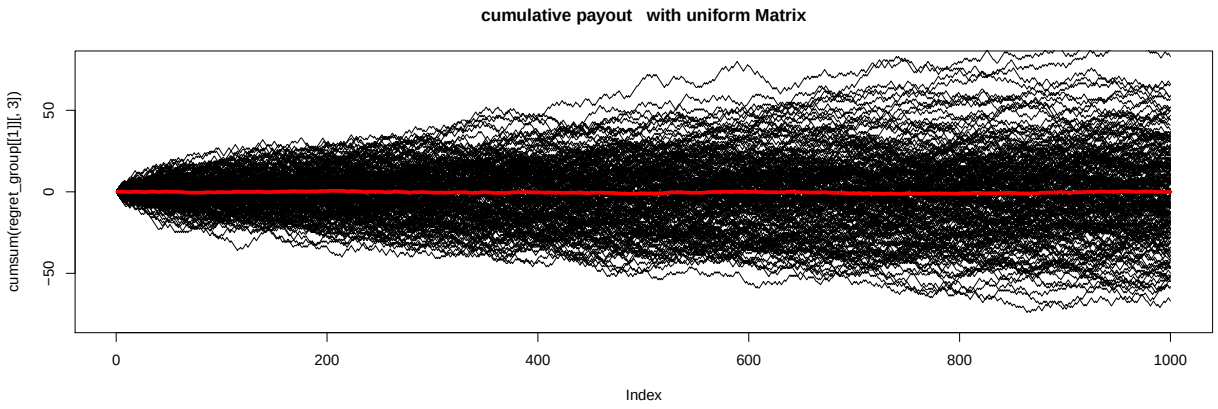
```
cum_pay_uniform(3)
```



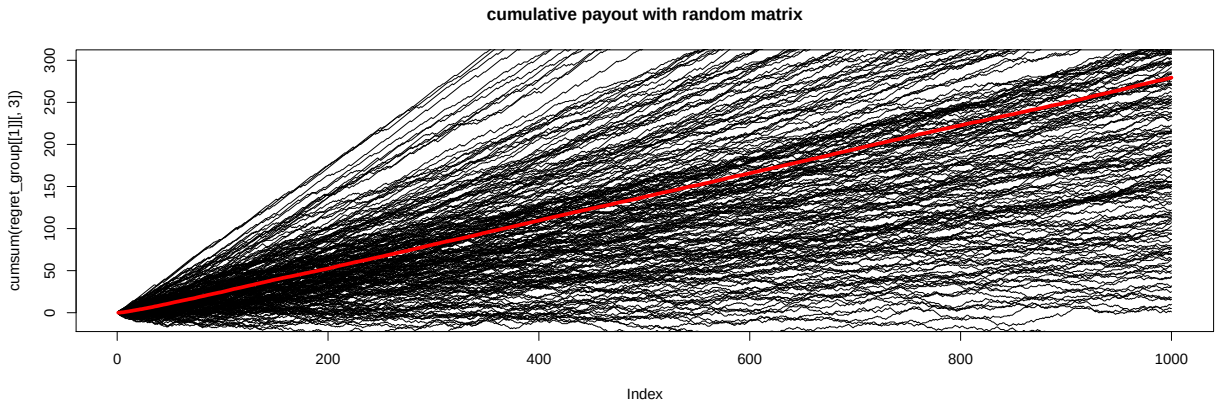

```
cum_pay_uniform(5)
```



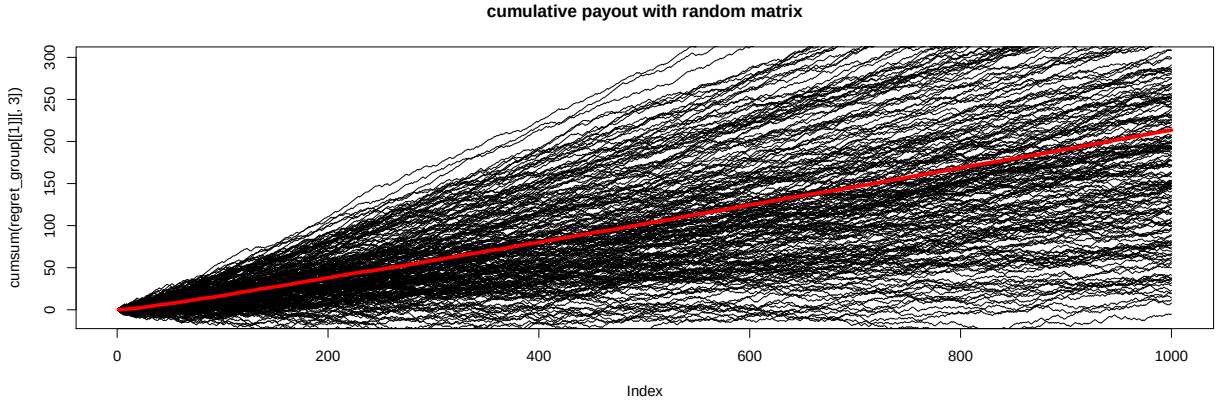

```
cum_pay_uniform(17)
```



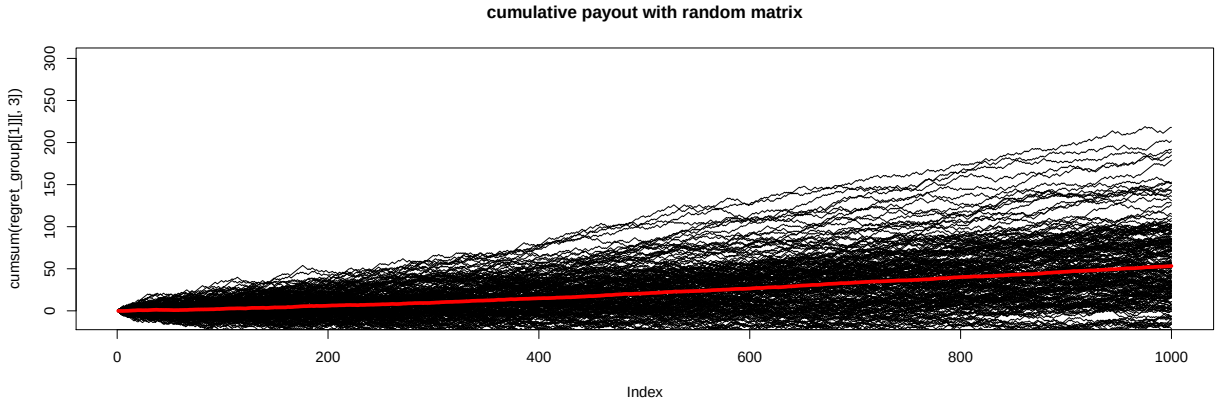
```
cum_pay_random(3)
```




```
cum_pay_random(5)
```




```
cum_pay_random(17)
```



I have created 3 different transition matrixes. First, I build a transition matrix A which is a identity matrix. As you may see in the plot name "cumulative payout with identity matrix", graph is increasing

exponentially. Also, in regrets plot, row player does not regret when he chooses 'paper' and 'scissors' because column player plays 'rock' all the time. Therefore, since row player is learning through my regret_match algorithm, cumulative payout graph is increasing exponentially.

However, when I have many independent sequences of actions for the column player using uniformly distributed transition matrix, A = 1/d(eeˆT), I cannot say that row player beats column players all the time. In the plot of 'cumulative payout with uniform matrix', the red line which is the average cum-plot of all realizations is a straight line at y = 0 approximately. Sometimes, the average plot is above 0 but sometimes that is under the 0. but averagely, the red line is at around y = 0.

Lastly, when I have a randomly generate the sequence of actions for the column player using random transition matrix, plot of "cumulative payout with random matrix" is keep increasing. my row-player is learning correctly, and he knows how to win against column-player which actions randomly by transition matrix. However, if I increase the dimension from 3 to 17, the slope of the average payout is decreasing. Finally, we will not able to say that row-player wind against column-player if and only if the dimension is high enough.

4. Explain the difference (if any) between the way your code is written and the exaple code, and what are teh advantages of they way you wrote yours.

I used different method for generating actions for the column-player. In the example code, Professor used Robin-Hood algorithm to generate the action for the the column player. However, I used markov chain with different versions of transtion matrix. I built transition matrix A = I, A = 1/d(eeˆT), and A = randomly generated transtion probability matrix. Also, as stated in the function of 'markov_y_identity', 'markov_y_uniform', and 'markov_y_random', I selected random vector for the random action at the first time step then multiplied by transtion matrix. As order increases, transtion matrix becomes matrix to the power of number of order. I just used the property of the markov chain. I did not use the fancy algorithm but I think this way generates the action correctly.