Econ4271_problem_set_1

Man Adrian Ho Hin

2023-02-27

```
rm(list=ls())
setwd("/Users/adrian/Desktop/econ4274_problem_set_1/")
library(lattice)
library(MASS)
```

Question 1a

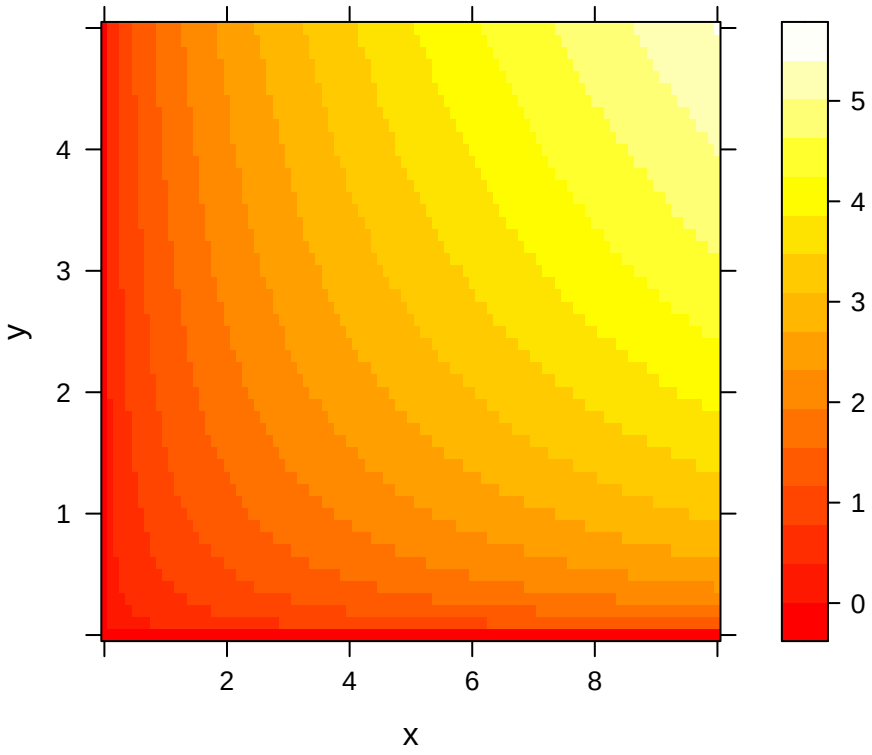
```
utility = function(x, y){
  return(x^(1/2)*y^(1/3))
}
p_x = 1
p_y = 2
w = 10
```

Question 1b

```
create_grid = function(upper_x, upper_y){
  x_seq = seq(0,upper_x,by=0.1)
  y_seq = seq(0,upper_y,by=0.1)
  xy_seq = NULL
  for (x in x_seq){
    for (y in y_seq){
      xy_seq = rbind(xy_seq, c(x,y))
    }
  }

  return(xy_seq)
}

temp = create_grid(w/p_x, w/p_y)
```

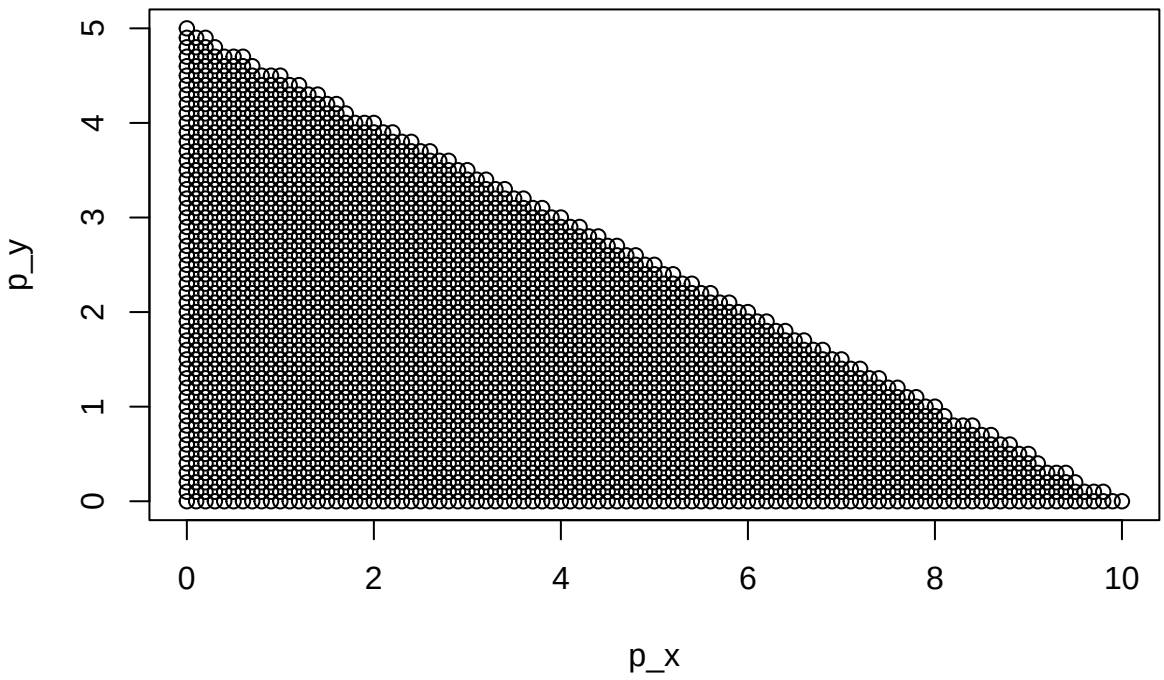
Question 1c

```
grid = create_grid(w/p_x,w/p_y)
useq = NULL
for(i in 1:length(grid[,1])){
  useq = rbind(useq, utility(grid[i,1], grid[i,2]))
}
grid = cbind(grid, useq)

plot(levelplot(grid[,3]~grid[,1]*grid[,2],
          col.regions=heat.colors(100),
          xlab = "x",
          ylab = "y",
          aspect=1)
     )
```



Question 1d

```
grid_bundles = grid[which(p_x*grid[,1]+p_y*grid[,2] <= w),]
plot(grid_bundles[,1], grid_bundles[,2],
     main = "Bundles that satisfy the budget constraint: p_x*x+p_y*y<=w",
     xlab = "p_x",
     ylab = "p_y")
```



Question 1e

```
colnames(grid_bundles) = c("x*", "y*", "v")
grid_bundles[which(grid_bundles[,3] == max(grid_bundles[,3])),]
```

```
##       x*       y*        v
## 6.000000 2.000000 3.086164
```

Question 1f

```
# a
UMP = function(u, p_x, p_y, w){
  # b
  grid = create_grid(w/p_x,w/p_y)

  # c
  useq = NULL
  for(i in 1:length(grid[,1])){
    useq = rbind(useq, u(grid[i,1], grid[i,2]))
  }
```

```
  grid = cbind(grid, useq)

  #plot(levelplot(grid[,3]~grid[,1]*grid[,2],
  #          col.regions=heat.colors(100),
  #          xlab = "x",
  #          ylab = "y",
  #          aspect=1)
  #     )

  # d
  grid_bundles = grid[which(p_x*grid[,1]+p_y*grid[,2] <= w),]
  #plot(grid_bundles[,1], grid_bundles[,2])

  # e
  outputs = grid_bundles[which(grid_bundles[,3] == max(grid_bundles[,3])),]

  return(outputs)
}
```

Question 1g

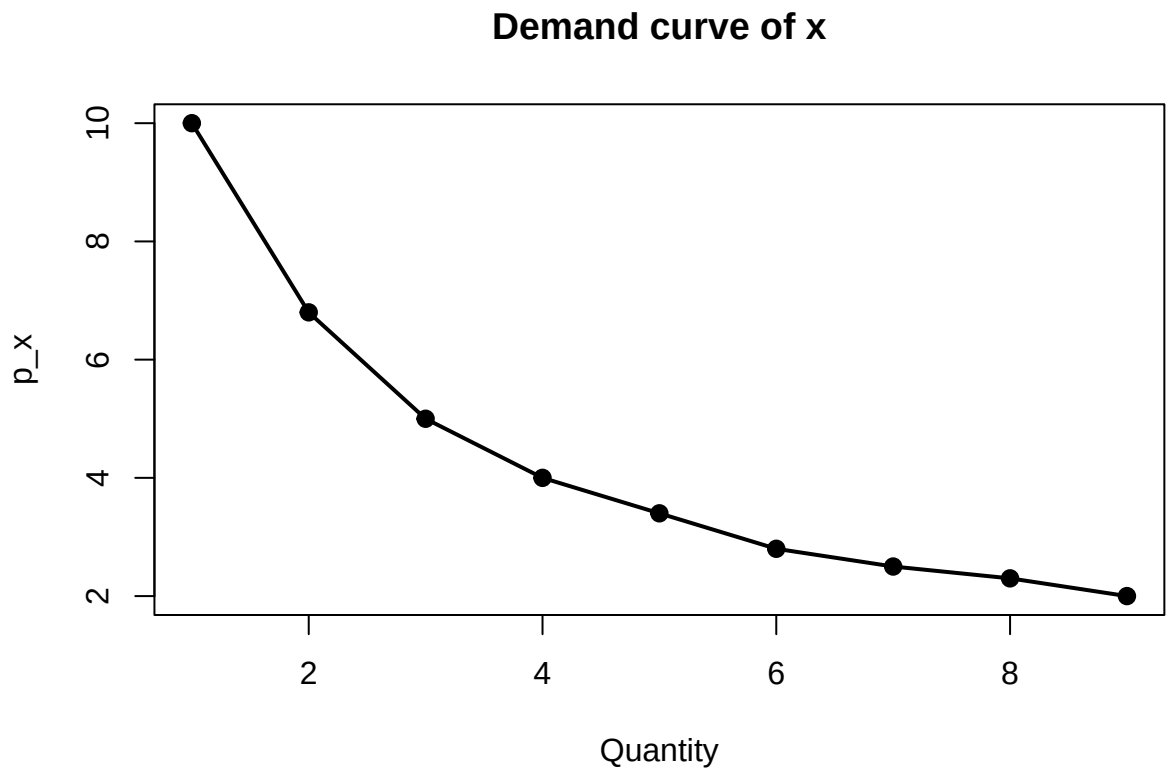
```
utility2 = function(x, y){
  return(x^(1/2)*y^(1/2))
}
p_x_2 = 4
p_y_2 = 2
w_2 = 20

results = UMP(utility2, p_x_2, p_y_2, w_2)
results = matrix(results,nrow=1)
colnames(results) = c("x*", "y*", "v")
results
```

```
##       x* y*        v
## [1,] 2.5  5 3.535534
```

Question 1h

```
X = NULL
utility3 = utility2
p_X = seq(1,5, by = 0.5)
p_y_3 = 2
w_3 = 20

for(i in p_X){
  X = c(X, UMP(utility3, i, p_y_3, w_3)[1])
}
```

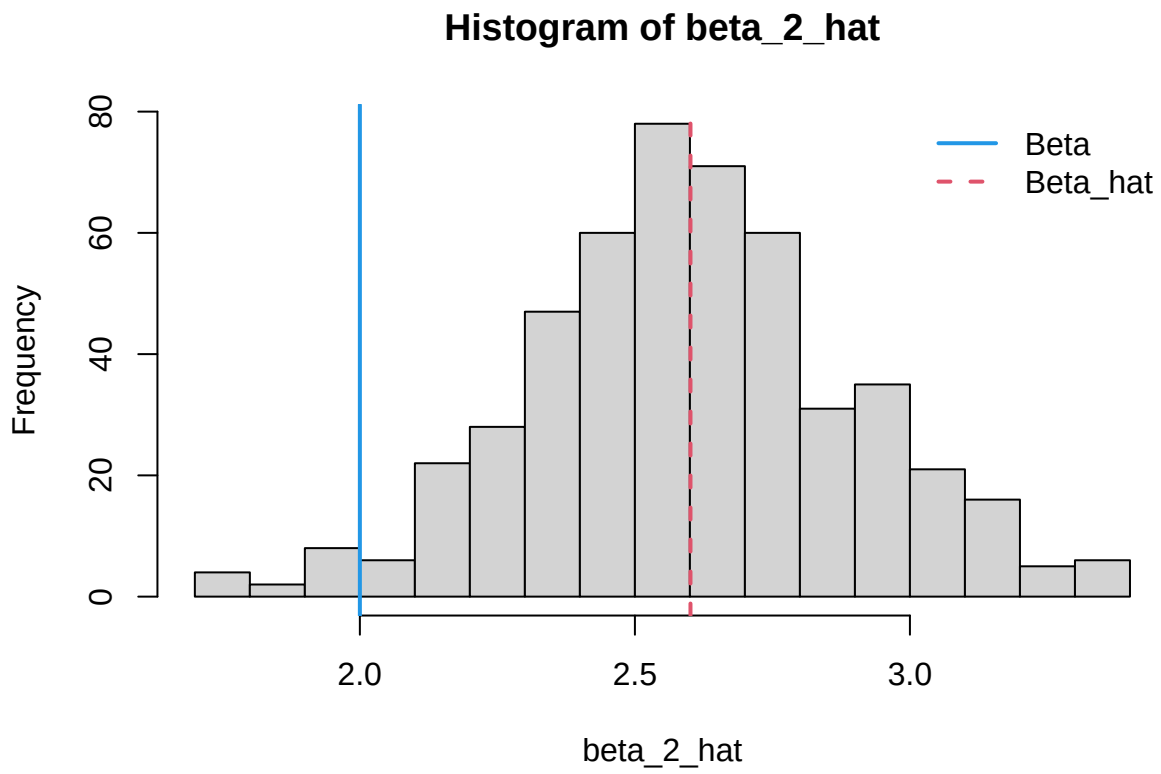
```
plot(X,
     main = "Demand curve of x",
     xlab = "Quantity",
     ylab = "p_x",
     lwd = 2,
     pch = 19)
lines(X, lwd = 2)
```



Question 2a

```
set.seed(1)
beta_1 = 1
beta_2 = 2
beta_3 = 3
n = 100
S = 500

beta_mat=NULL
for (s in 1:S){
  x_0 = rep(1,n)
  X = mvrnorm(n, c(2,1), matrix(c(1,0.2,0.2,1), 2,2))
  x_1 = X[,1]
```

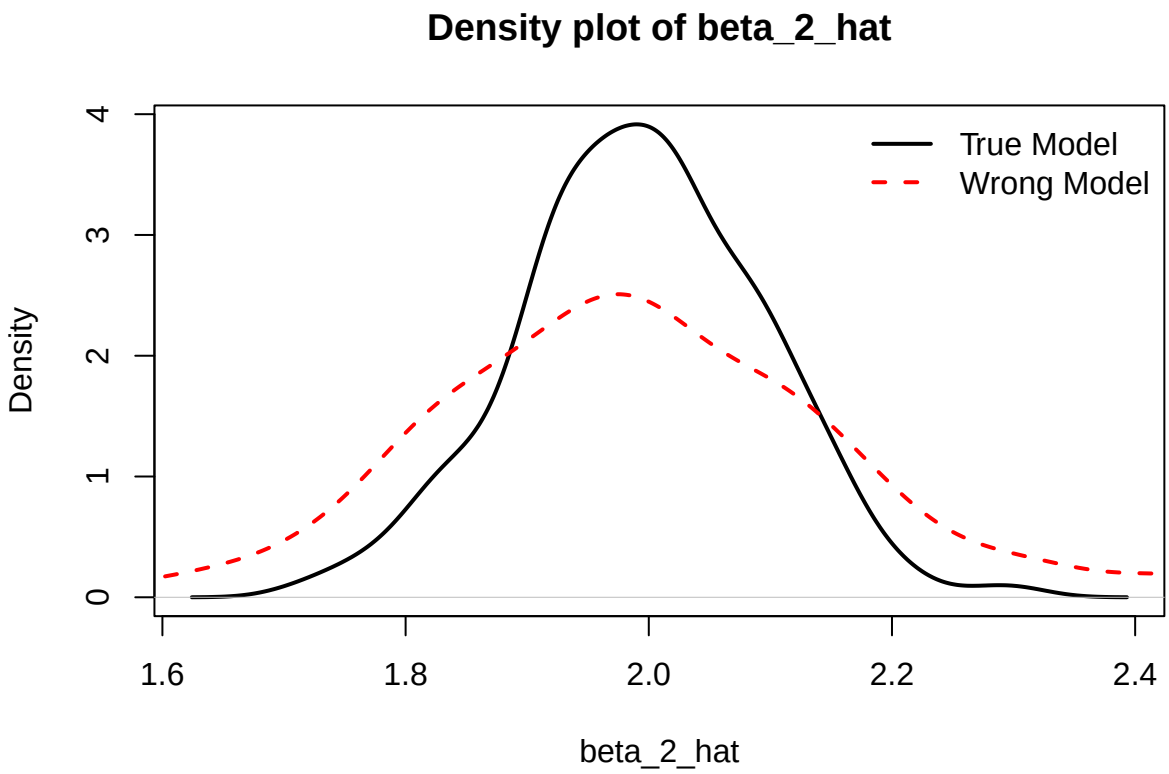
```
  x_2 = X[,2]
  e = rnorm(n,0,1)

  # true model
  y = beta_1*x_0+beta_2*x_1+beta_3*x_2+e
  # omitted model
  lm = lm(y~x_1)

  beta_mat=rbind(beta_mat,lm$coefficients)
}

hist(beta_mat[,2],
     breaks=20,
     main = "Histogram of beta_2_hat",
     xlab = "beta_2_hat")
abline(v=beta_2,col=4,lwd=2)
abline(v=mean(beta_mat[,2]),col=2,lwd=2,lty=2)
legend("topright",
       c("Beta","Beta_hat"),
       bty="n",
       col=c(4,2),lty=1:2,lwd=2)
```



It over estimated the Beta on average, the beta is unlikely to be estimated correctly.

Question 2b

```
set.seed(1)
beta_1 = 1
beta_2 = 2
beta_3 = 3
n = 100
S = 500

# true model
beta_mat_true = NULL
beta_mat_wrong = NULL
for (s in 1:S){
  x_0 = rep(1,n)
  X = mvrnorm(n, c(2,1), matrix(c(1,0.8,0.8,1), 2,2))
  x_1 = X[,1]
  x_2 = X[,2]
  u = rnorm(n,0,1)

  # true model
  y = beta_1*x_0+beta_2*x_1+u
  lm_true = lm(y~x_1)
  # wrong model
  lm_wrong = lm(y~x_1+x_2)

  beta_mat_true = rbind(beta_mat_true,lm_true$coefficients)
  beta_mat_wrong = rbind(beta_mat_wrong, lm_wrong$coefficients)
}

plot(density(beta_mat_true[,2]),
     main = "Density plot of beta_2_hat",
     xlab = "beta_2_hat",
     col="black",
     lwd=2)
lines(density(beta_mat_wrong[,2]),
      col="red",
      lty=2,
      lwd=2)
legend("topright",
       c("True Model","Wrong Model"),
       bty="n",
       col=c("black","red"),lty=1:2,lwd=2)
```



The wrong model has a higher standard error for the beta_2_hat than the beta in the true model.

Question 2c

The null hypothesis H_0: beta_3 = 0

```
# part (b) simulation
results_b = NULL
for (s in 1:S){
  x_0 = rep(1,n)
  X = mvrnorm(n, c(2,1), matrix(c(1,0.8,0.8,1), 2,2))
  x_1 = X[,1]
  x_2 = X[,2]
  u = rnorm(n,0,1)
  y = beta_1*x_0+beta_2*x_1+u

  lm_restricted = lm(y~x_1)
  lm_unrestricted = lm(y~x_1+x_2)

  result = anova(lm_restricted,lm_unrestricted)[[6]][2]

  results_b = c(results_b, result)
}

cat("The Type I error rate: " , length(which(results_b < 0.05))/S*100, "%\n")
```

```
## The Type I error rate:  4.2 %
```

```
# part (a) simulation
results_a = NULL
for (s in 1:S){
  x_0 = rep(1,n)
  X = mvrnorm(n, c(2,1), matrix(c(1,0.2,0.2,1), 2,2))
  x_1 = X[,1]
  x_2 = X[,2]
  e = rnorm(n,0,1)
  y = beta_1*x_0+beta_2*x_1+beta_3*x_2+e

  lm_restricted = lm(y~x_1)
  lm_unrestricted = lm(y~x_1+x_2)

  result = anova(lm_restricted,lm_unrestricted)[[6]][2]

  results_a = c(results_a, result)
}

cat("The Type II error rate: " , length(which(results_a > 0.05))/S, "%\n")
```

```
## The Type II error rate:  0 %
```

Question 3a

```
player1_payoff = function(action1, action2){
  payoff_matrix = array(data = c(c(0,1,-1),c(-1,0,1),c(1,-1,0)), dim = c(3,3))
  return(payoff_matrix[action1,action2])
}
```

Question 3b

```
player1_payoff_prob = function(r1,p1,r2,p2){
  expected_utility = r1*(-p2+(1-r2-p2))+p1*(r2-(1-r2-p2))+(1-r1-p1)*(-r2+p2)
  return(expected_utility)
}

player1_payoff_prob(0.1,0.1,0.8,0.1)
```

```
## [1] -0.49
```

Question 3c

```
player_action = function(r,p){
  action = sample(c(1,2,3),1,prob=c(r,p,1-r-p))
  return(action)
}

player2_actions = NULL
for (i in 1:5000){
  player2_action = player_action(0.4,0.3)
  player2_actions = c(player2_actions, player2_action)
}
```

Question 3d

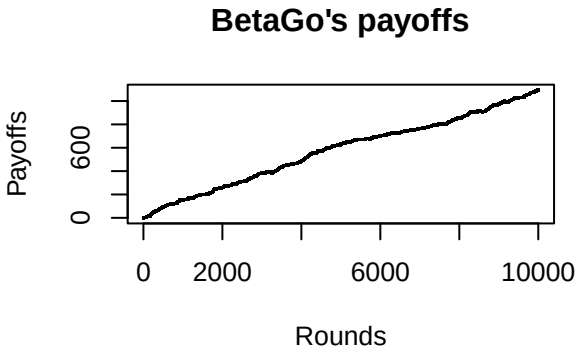
```
BetaGo = function(opponent_actions, beginning = TRUE){
  if (beginning == FALSE){
    predicted_r = length(which(opponent_actions==1))/length(opponent_actions)
    predicted_p = length(which(opponent_actions==2))/length(opponent_actions)
    if(predicted_r>1/3|predicted_p>1/3|(1-predicted_r-predicted_p)>1/3){
      utilities = NULL
      for (i in 1:length(total_seq[,1])){
        payoff = player1_payoff_prob(total_seq[i,1], total_seq[i,2], r2, p2)
        utilities = c(utilities, round(payoff, 2))
      }
      best_action = total_seq[which(utilities == max(utilities)),]

      action = player_action(best_action[1], best_action[2])
      return(action)
    }
  }
  action = player_action(1/3,1/3)
  return(action)
}

r_seq = seq(0,1,by=0.1)
p_seq = seq(0,1,by=0.1)
total_seq = NULL
for (r in r_seq){
  for (p in p_seq){
    if (r+p <= 1){
      total_seq = rbind(total_seq, c(r,p))
    }
  }
}

## simulation
r2 = 0.4
p2 = 0.3
player2_actions = NULL
player1_payoffs = NULL
BetaGo_payoffs = NULL
```

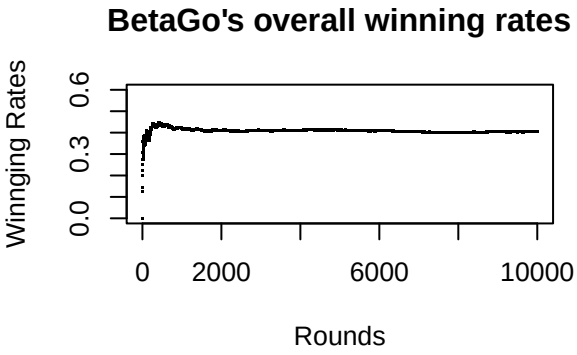
```
player1_winnings = NULL
player1_drawings = NULL
player1_lossings = NULL
set.seed(4274)

i = 0
rounds = c(1, 10, 100, 1000, 10000)
while (i != rounds[length(rounds)]){
  i = i+1

  player2_action = player_action(r2,p2)
  if (i <= 10){
    player1_payoffs = c(player1_payoffs,
                        player1_payoff(
                          BetaGo(player2_actions),
                          player2_action))
  }
  else{
    player1_payoffs = c(player1_payoffs,
                        player1_payoff(
                          BetaGo(player2_actions, beginning = FALSE),
                          player2_action))
  }

  player2_actions = c(player2_actions, player2_action)
  BetaGo_payoffs = c(BetaGo_payoffs, sum(player1_payoffs))
  player1_winnings = c(player1_winnings, length(which(player1_payoffs==1))/i)
  player1_drawings = c(player1_drawings, length(which(player1_payoffs==0))/i)
  player1_lossings = c(player1_lossings, length(which(player1_payoffs==-1))/i)
  if(i %in% rounds){
    cat("BetaGo after", i, "rounds of simulation: \n")
    cat("   Payoffs: ", sum(player1_payoffs), "\n")
    cat("   Winnging rate: ", length(which(player1_payoffs==1))/i, "\n")
    cat("   Drawing rate: ", length(which(player1_payoffs==0))/i, "\n")
    cat("   Lossing rate: ", length(which(player1_payoffs==-1))/i, "\n")
  }
}
```

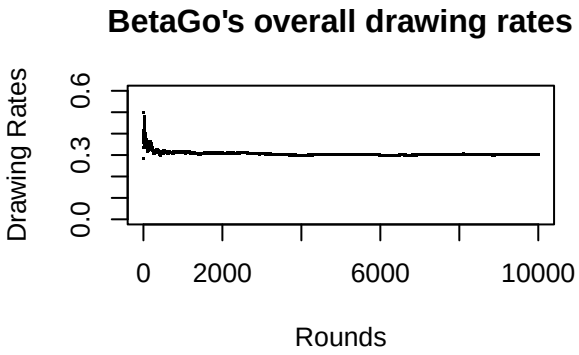
```
## BetaGo after 1 rounds of simulation:
##    Payoffs:  0
##    Winnging rate:  0
##    Drawing rate:  1
##    Lossing rate:  0
## BetaGo after 10 rounds of simulation:
##    Payoffs:  -2
##    Winnging rate:  0.2
##    Drawing rate:  0.4
##    Lossing rate:  0.4
## BetaGo after 100 rounds of simulation:
##    Payoffs:  14
##    Winnging rate:  0.4
##    Drawing rate:  0.34
##    Lossing rate:  0.26
```

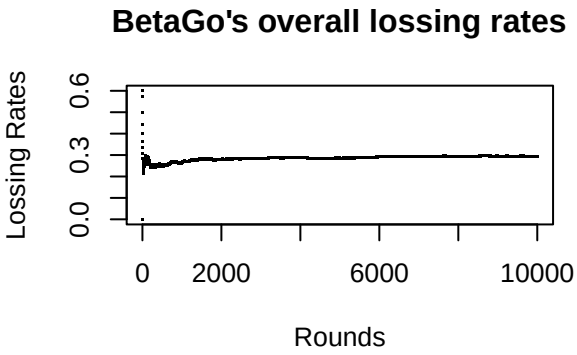
```
## BetaGo after 1000 rounds of simulation:
##    Payoffs:  153
##    Winnging rate:  0.419
##    Drawing rate:  0.315
##    Lossing rate:  0.266
## BetaGo after 10000 rounds of simulation:
##    Payoffs:  1096
##    Winnging rate:  0.4042
##    Drawing rate:  0.3012
##    Lossing rate:  0.2946
```

```
par(mfcol = c(2, 2))
plot(BetaGo_payoffs, pch=".",
     main = "BetaGo's payoffs",
     xlab = "Rounds",
     ylab = "Payoffs")

plot(player1_drawings, pch=".",
     main = "BetaGo's overall drawing rates",
     xlab = "Rounds",
     ylab = "Drawing Rates",
    ylim = c(0,0.6))

plot(player1_winnings, pch=".",
     main = "BetaGo's overall winning rates",
     xlab = "Rounds",
     ylab = "Winnging Rates",
     ylim = c(0,0.6))

plot(player1_lossings, pch=".",
     main = "BetaGo's overall lossing rates",
     xlab = "Rounds",
     ylab = "Lossing Rates",
    ylim = c(0,0.6))
```


BetaGo's payoffs
Payoffs
0
600
0
2000
6000
10000
Rounds


BetaGo's overall winning rates
Winnging Rates
0.0
0.3
0.6
0
2000
6000
10000
Rounds


BetaGo's overall drawing rates
Drawing Rates
0.0
0.3
0.6
0
2000
6000
10000
Rounds


BetaGo's overall lossing rates
Lossing Rates
0.0
0.3
0.6
0
2000
6000
10000
Rounds