Assignment2

Michael Seebregts

2025-10-27

```
## Warning: package 'magick' was built under R version 4.5.1
```

```
## Linking to ImageMagick 6.9.13.29
## Enabled features: cairo, freetype, fftw, ghostscript, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fontconfig, x11
```

#Q1

Q1a

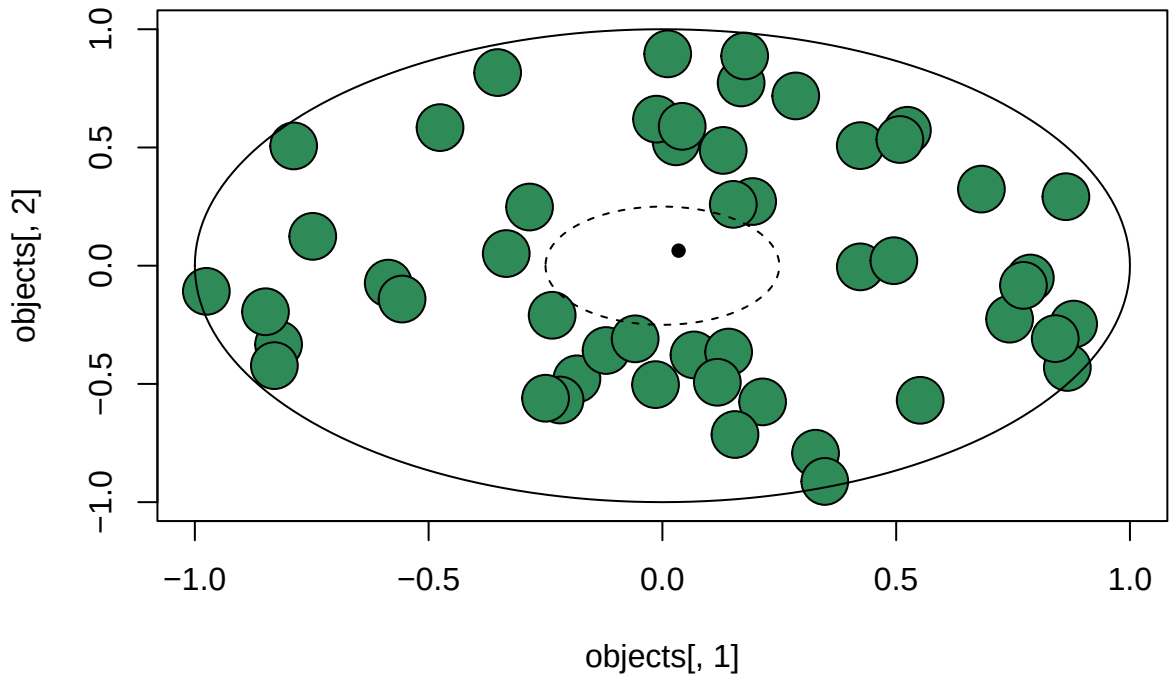
```
game_status=function(xt,objects, rad)
{
  sk    = rep(0,dim(xt)[1])
  min_dists = rep(0,dim(xt)[1])
  J         = dim(objects)[1]
  ones      = matrix(1,J,1)
  for(i in 1:dim(xt)[1])
  {
    min_dists[i] = min(sqrt(rowSums((objects-ones%*%xt[i,])^2)))

    if (((xt[i, 1])^2 + (xt[i, 2]^2 ) >= 1))
    {
      sk[i] = 1
    }
    else if (min_dists[i] < rad)
    {
      sk[i] = -1
    }
    else
    {
      sk[i] = 0
    }
  }

  #sk = 1*(xt[,2]>1)-1*((xt[,1]< -1)|(xt[,1]> +1)|(xt[,2]< -1)|(min_dists<rad))
  ret = list(status = sk, minDist = min_dists)
  return(ret)
}
```

Q1b

i

```
options(scipen = 999)

draw_objects = function(J)
{
  r_o       =  runif(J,0.3,1)
  pi_vals   =  runif(J,-1,1)*pi
  o1        =  r_o*cos(pi_vals)     # x-coordinates of objects
  o2        =  r_o*sin(pi_vals)
  return(cbind(o1,o2))
}
draw_starts = function(N = 1)
{
  r_x      =  runif(N,0,0.25)
```

```
  pi_x     =  runif(N,-1,1)*pi
  xt       =  matrix(cbind(r_x*cos(pi_x),r_x*sin(pi_x)),nrow = N,byrow = TRUE)
}

set.seed(2024)
J       =  50
objects = draw_objects(J)
xt      = draw_starts()
delt    = 0.025              # How big are the steps you can take?
rad     = 0.05               # How close to an object before you crash?

# Just a function to plot the game:
plot_game = function(xt,objects,rad,cols = 1)
{
  plot(objects[,2]~objects[,1],type = 'n', ylim = c(-1,1), xlim = c(-1,1))
  symbols(objects[,1],objects[,2],circles = rep(rad,J),ylim = c(-1,1),xlim = c(-1,1),inches = FALSE,add
  points(xt[,2]~xt[,1], pch = 16, cex = 1,col = cols)
  pi_v = seq(-1,1,1/100)*pi
  y_edge = sin(pi_v)
  x_edge = cos(pi_v)
  lines(y_edge~x_edge)
  lines(c(0.25*y_edge)~c(0.25*x_edge),lty = 2)
}

plot_game(xt,objects,rad,'black')
```



```
play = function(x0,delt,objects,rad,theta,plt = FALSE,trace = FALSE)
{
  k           =   0 # Count how many steps
  xt          = x0   # Set the initial coordinate(s) for the drone.
  trajectories = NULL
  # Check the game status:
  res_status  = game_status(xt, objects, rad)
  status      = res_status$status
    # Check which games are still active:
  terminal      = (status != 0)
  if(trace)
  {
    trajectories = array(dim = c(dim(xt)[1],dim(xt)[2],101))
    trajectories[,,1] = xt
  }
  if(plt){plot_game(xt,objects,rad,'black')}
  while((any(status==0))&(k<100))
  {
    k = k + 1

    # Now, let the model update the position of the pieces:

    ct = control(xt, theta)
    xt = xt+ct*delt*cbind(1-terminal,1-terminal)

    # Checkk the game status after the positions are updates:
    res_status  = game_status(xt, objects, rad)
    status      = res_status$status
    terminal    = (status != 0)
    if(trace){trajectories[,,k] = xt}
    if(plt){plot_game(xt,objects,rad,c('red','black','green')[status+2])}
  }
  return(list(k = k, status = status,xt= xt,trajectories = trajectories))
}


model = function(X,theta,nodes)
{
  # Infer dimensions:
  N = dim(X)[1]
  p = dim(X)[2]
  q = 2
  dims = c(p,nodes,q)

  # Populate weight and bias matrices:
  index = 1:(dims[1]*dims[2])
  W1    = matrix(theta[index],dims[1],dims[2])
  index = max(index)+1:(dims[2]*dims[3])
  W2    = matrix(theta[index],dims[2],dims[3])
  index = max(index)+1:(dims[3]*dims[4])
  W3    = matrix(theta[index],dims[3],dims[4])

  index = max(index)+1:(dims[2])
```

```
  b1    = matrix(theta[index],dims[2],1)
  index = max(index)+1:(dims[3])
  b2    = matrix(theta[index],dims[3],1)
  index = max(index)+1:(dims[4])
  b3    = matrix(theta[index],dims[4],1)

  ones    = matrix(1, 1, N)
  a0      = t(X)

    # Evaluate the updating equation in matrix form
  a1 = tanh(t(W1)%*%a0 + b1%*%ones)
  a2 = tanh(t(W2)%*%a1 + b2%*%ones)
  a3 = tanh(t(W3)%*%a2 + b3%*%ones)

  # Return a list of relevant objects:
  return(list(a3 = t(a3)))
}

p     = 2
q     = 2
nodes = 3
npars = p*nodes+nodes*nodes+nodes*q+nodes+nodes+q
npars
```

```
## [1] 29
```

```
theta_rand = runif(npars,-1,1)

control = function(xt,pars)
{

  res_model = model(xt,pars,rep(nodes,2))
  return(res_model$a3)
}
control(xt,theta_rand)
```

```
##            [,1]      [,2]
## [1,] 0.06081162 0.3134288
```

```
play_a_game = function(theta)
{
  xt    = draw_starts()
  res   = play(xt,delt,objects,rad,theta, trace = TRUE)
  score = mean(res$status==1)
  return(score)
}
# play_a_game(theta_rand)

library('GA')
```

```
## Warning: package 'GA' was built under R version 4.5.1
```

```
## Loading required package: foreach

## Loading required package: iterators

## Package 'GA' version 3.2.4
## Type 'citation("GA")' for citing this R package in publications.

##
## Attaching package: 'GA'

## The following object is masked from 'package:utils':
##
##     de
```

```
obj = play_a_game
GA  = ga(type = 'real-valued',fitness = play_a_game,lower = rep(-10,npars),upper = rep(10,npars), popSi
#plot(GA)


theta_hat = GA@solution[1,]
#theta_hat

fps = 1
w = 800; h = 800
img = image_graph(width = w, height = h, res = 120)

win = c()
for(i in 1:100)
{

  #objects = draw_objects(J)
  xt_try    = draw_starts()
  plot_game(xt_try,objects,rad,'black')
  res_final = play(xt_try,delt,objects,rad,theta_hat,plt = FALSE,trace = TRUE)
  #print(typeof(res_final$trajectories))

  trajs = na.omit(res_final$trajectories)
  lines(trajs[, 2,]~trajs[, 1, ])
  win[i] = (res_final$status==1)
}
propWins = data.frame("Proportion of successful navigations" = mean(win)*100)
kable(propWins)
```

Proportion.of.successful.navigations
48

ii

```
img2 = image_graph(width = w, height = h, res = 120)
winNewObjs = c()
for(i in 1:100)
{

  objects = draw_objects(J)
  xt_try    = draw_starts()
  plot_game(xt_try,objects,rad,'black')
  res_final = play(xt_try,delt,objects,rad,theta_hat,plt = FALSE,trace = TRUE)
  #print(typeof(res_final$trajectories))

  trajs = na.omit(res_final$trajectories)
  lines(trajs[, 2,]~trajs[, 1, ])
  winNewObjs[i] = (res_final$status==1)
}
propNewWins = data.frame("Proportion of successful navigations" = mean(winNewObjs))
kable(propNewWins)
```

Proportion.of.successful.navigations
0.27

Q1c

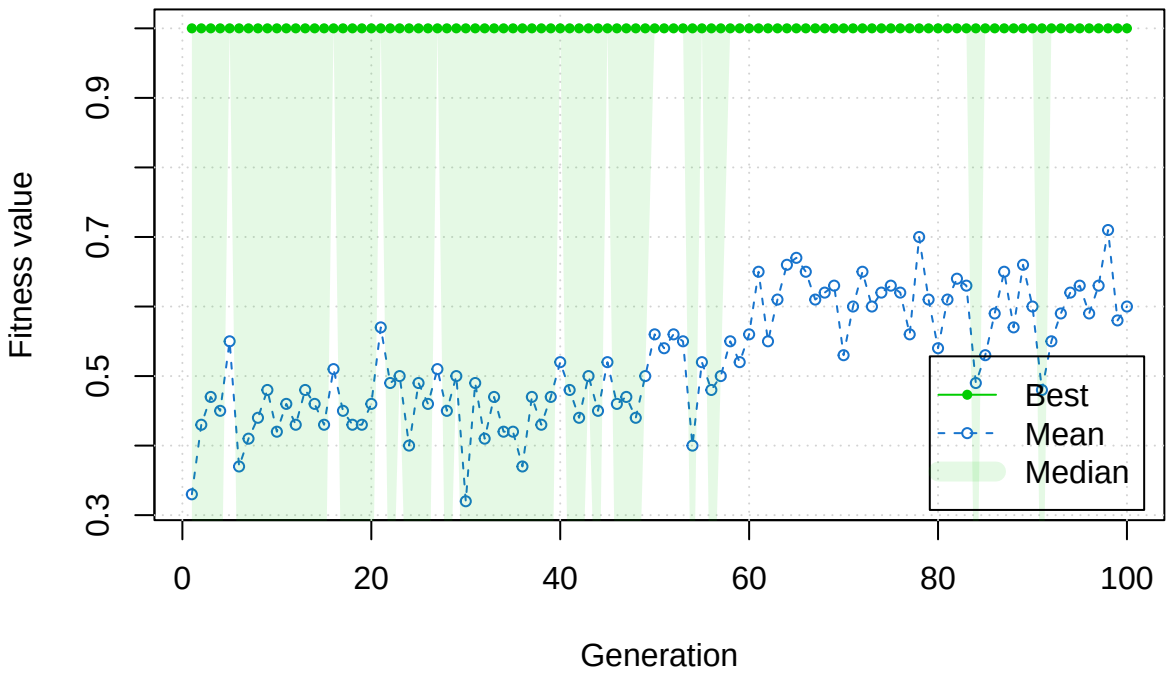
i

```
### c

playAdj = function(x0,delt,objects,rad,theta,plt = FALSE,trace = FALSE)
{
  k           =   0 # Count how many steps
  xt          = x0   # Set the initial coordinate(s) for the drone.
  trajectories = NULL
  # Check the game status:
  res_status  = game_status(xt, objects, rad)
  status      = res_status$status
  minDists = res_status$minDist

  inverseDist = 1/(minDists-rad)
  # Check which games are still active:
  terminal      = (status != 0)
  if(trace)
  {
    trajectories = array(dim = c(dim(xt)[1],dim(xt)[2],101))
    trajectories[,,1] = xt
  }
  if(plt){plot_game(xt,objects,rad,'black')}
  while((any(status==0))&(k<100))
  {
    k = k + 1
```

```
    # Now, let the model update the position of the pieces:

    ct = controlAdjusted(xt, theta, inverseDist)
    xt = xt+ct*delt*cbind(1-terminal,1-terminal)

    # Checkk the game status after the positions are updates:
    res_status  = game_status(xt, objects, rad)
    status      = res_status$status
    minDists = res_status$minDist
    inverseDist = 1/(minDists-rad)
    terminal    = (status != 0)
    if(trace){trajectories[,,k] = xt}
    if(plt){plot_game(xt,objects,rad,c('red','black','green')[status+2])}
  }
  return(list(k = k, status = status,xt= xt,trajectories = trajectories))
}


controlAdjusted = function(xt,pars, inverseDist)
{
  xt_aug = cbind(xt, inverseDist)
  res_model = model(xt_aug,pars,rep(nodes,2))
  return(res_model$a3)
}

play_a_game_adj = function(theta)
{
  xt    = draw_starts()
  res   = playAdj(xt,delt,objects,rad,theta, trace = TRUE)
  score = mean(res$status==1)
  return(score)
}

p     = 3
q     = 2
nodes = 3
npars = p*nodes+nodes*nodes+nodes*q+nodes+nodes+q
npars
```

```
## [1] 32
```

```
theta_rand = runif(npars,-1,1)

objAdj = play_a_game_adj
GA_adj  = ga(type = 'real-valued',fitness = play_a_game_adj,lower = rep(-10,npars),upper = rep(10,npars)
plot(GA_adj)
```




```
theta_hat_adj = GA_adj@solution[1,]
theta_hat_adj
```

```
##         x1         x2         x3         x4         x5         x6         x7
## -0.9701872  0.7625378  2.2235240 -0.4363090  3.3083093 -0.9100121 -1.8116587
##         x8         x9        x10        x11        x12        x13        x14
## -0.6714361  2.2081672 -0.8227137  1.8983000 -0.2765110 -3.1899070 -0.2120395
##        x15        x16        x17        x18        x19        x20        x21
##  2.1475885  1.5860119 -2.0250661 -0.0130785 -1.0213971  2.8699384 -1.0138585
##        x22        x23        x24        x25        x26        x27        x28
##  1.8663929 -1.1554169 -0.3377329 -1.4882045  7.7751805 -1.0679383  3.0472488
##        x29        x30        x31        x32
##  0.4254153  0.6348349  1.1963050  3.2648460
```

```
img3 = image_graph(width = w, height = h, res = 120)
win_adj = c()
for(i in 1:100)
{

  #objects = draw_objects(J)
  xt_try    = draw_starts()
  plot_game(xt_try,objects,rad,'black')
  res_final = playAdj(xt_try,delt,objects,rad,theta_hat_adj,plt = FALSE,trace = TRUE)
  #print(typeof(res_final$trajectories))
```

```
  trajs = na.omit(res_final$trajectories)
  lines(trajs[, 2,]~trajs[, 1, ])
  win_adj[i] = (res_final$status==1)
}

propWinsAdj = data.frame("Proportion of successful navigations" = mean(win_adj)*100)
kable(propWinsAdj)
```

Proportion.of.successful.navigations
29

```
img4 = image_graph(width = w, height = h, res = 120)
winNewObjs_adj = c()
for(i in 1:100)
{

  objects = draw_objects(J)
  xt_try    = draw_starts()
  plot_game(xt_try,objects,rad,'black')
  res_final = playAdj(xt_try,delt,objects,rad,theta_hat_adj,plt = FALSE,trace = TRUE)
  #print(typeof(res_final$trajectories))

  trajs = na.omit(res_final$trajectories)
  lines(trajs[, 2,]~trajs[, 1, ])
  winNewObjs_adj[i] = (res_final$status==1)
}
mean(winNewObjs_adj)
```

```
## [1] 0.28
```

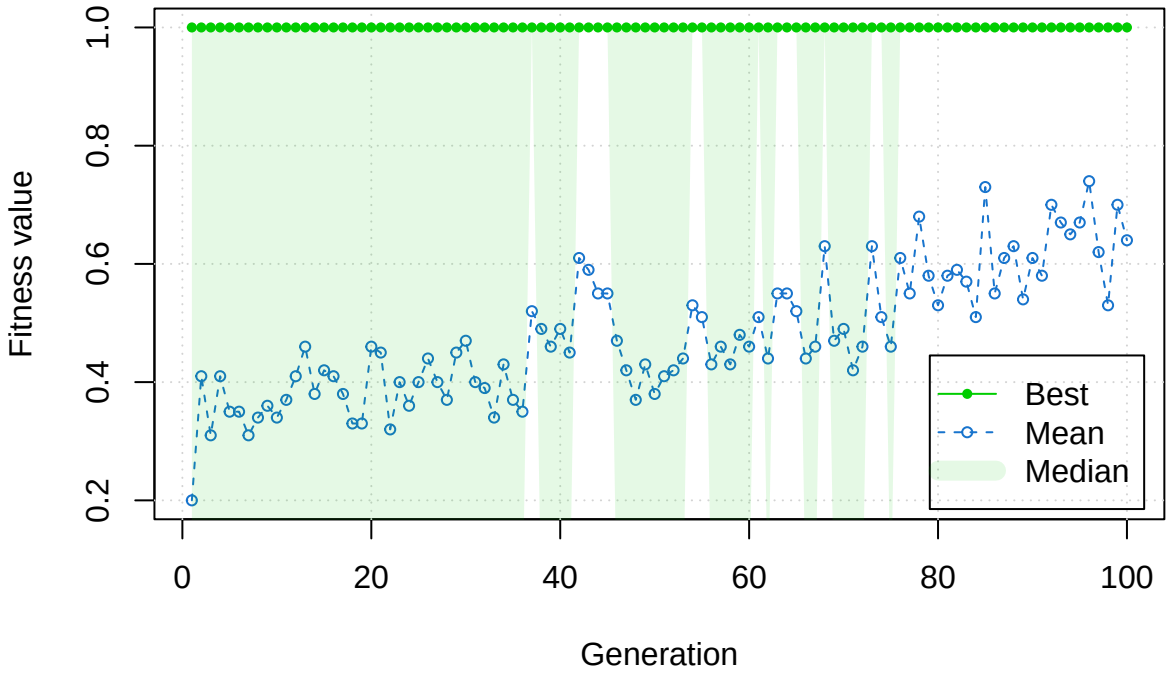
```
propNewWinsAdj = data.frame("Proportion of successful navigations" = mean(winNewObjs_adj)*100)
kable(propNewWinsAdj)
```

Proportion.of.successful.navigations
28

Q1d

```
### d
play_a_game_adj_newObs = function(theta)
{
  xt    = draw_starts()
  objects = draw_objects(J)
  res   = playAdj(xt,delt,objects,rad,theta, trace = TRUE)
  score = mean(res$status==1)
  return(score)
}
```

```
objAdj = play_a_game_adj_newObs
GA_adj_newObs  = ga(type = 'real-valued',fitness = play_a_game_adj_newObs,lower = rep(-10,npars),upper =
plot(GA_adj_newObs)
```




```
theta_hat_adj_newObs = GA_adj_newObs@solution[1,]
theta_hat_adj_newObs
```

```
##         x1         x2         x3         x4         x5         x6         x7
##  3.9029900  3.8203046  0.3020354  3.6447905  3.4153999 -1.6367394  1.5385434
##         x8         x9        x10        x11        x12        x13        x14
##  1.3441913 -0.4753239 -0.7238650  3.0368669  0.2852006 -0.6652189  1.2127239
##        x15        x16        x17        x18        x19        x20        x21
##  2.0225949  2.7806615 -0.3843214  2.9709117  0.5513355  0.9206854  4.1124271
##        x22        x23        x24        x25        x26        x27        x28
## -0.4281843  0.7484773 -1.5721754 -2.1998529  0.8128878 -0.7787385  2.9601865
##        x29        x30        x31        x32
## -3.4875263  1.9224238 -1.9864429 -0.4461203
```

```
img5 = image_graph(width = w, height = h, res = 120)
win_adj_newObs = c()
for(i in 1:100)
{
```

```
  #objects = draw_objects(J)
  xt_try    = draw_starts()
  plot_game(xt_try,objects,rad,'black')
  res_final = playAdj(xt_try,delt,objects,rad,theta_hat_adj_newObs,plt = FALSE,trace = TRUE)
  #print(typeof(res_final$trajectories))

  trajs = na.omit(res_final$trajectories)
  lines(trajs[, 2,]~trajs[, 1, ])
  win_adj_newObs[i] = (res_final$status==1)
}
mean(win_adj_newObs)
```

```
## [1] 0.56
```

```
img6 = image_graph(width = w, height = h, res = 120)
winNewObjs_adj_genNewObs = c()
for(i in 1:100)
{

  objects = draw_objects(J)
  xt_try    = draw_starts()
  plot_game(xt_try,objects,rad,'black')
  res_final = playAdj(xt_try,delt,objects,rad,theta_hat_adj_newObs,plt = FALSE,trace = TRUE)
  #print(typeof(res_final$trajectories))

  trajs = na.omit(res_final$trajectories)
  lines(trajs[, 2,]~trajs[, 1, ])
  winNewObjs_adj_genNewObs[i] = (res_final$status==1)
}
mean(winNewObjs_adj_genNewObs)
```

```
## [1] 0.6
```

Q2

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
##  [1,]    1    0    0    1    0    0    1    0
##  [2,]    1    0    0    0    1    0    0    0
##  [3,]    1    0    0    0    0    1    0    1
##  [4,]    0    1    0    1    0    0    0    0
##  [5,]    0    1    0    0    1    0    1    1
##  [6,]    0    1    0    0    0    1    0    0
##  [7,]    0    0    1    1    0    0    0    1
##  [8,]    0    0    1    0    1    0    0    0
##  [9,]    0    0    1    0    0    1    1    0
```

Q2a

```
game_tree= function(m,k, currMoves = 0, maxMoves = 9)
{
  g = c()
  game_state = rho(m,S)
  if(game_state$terminal)
  {
    g = c(g,game_state$winner)
    return(g)
  }
  else if(currMoves == maxMoves)
  {
    g = c(g, 2)
    return(g)
  }

  else{
    Index = which(m == 0)
    for(i in 1:length(Index))
    {
      x = m
      x[Index[i]]=k
      g = c(g,game_tree(x,-1*k, currMoves + 1, maxMoves))
    }
    return(g)
  }
}

m = as.matrix(c(0,0,0,0,0,0,0,0,0))

res5 = game_tree(m,1, 0, 5)
n_g5  =     length(res5)
Xwins5 = sum(res5==-1)
Draws5 = sum(res5== 0)
Owins5 = sum(res5==+1)
Unfinished5 = sum(res5==+2)
#Games5 = c(n_g5,Xwins5,Draws5,Owins5, Unfinished5)
games5 = data.frame("Number of moves" = 5, "Number of games" = n_g5, "X wins" = Xwins5, "O wins" = Owins

res8 = game_tree(m,1, 0, 8)
n_g8  =     length(res8)
Xwins8 = sum(res8==-1)
Draws8 = sum(res8== 0)
Owins8 = sum(res8==+1)
Unfinished8 = sum(res8==+2)
#Games8 = c(n_g8,Xwins8,Draws8,Owins8, Unfinished8)
games8 = data.frame("Number of moves" = 8, "Number of games" = n_g5, "X wins" = Xwins5, "O wins" = Owins

res9 = game_tree(m,1, 0, 9)
n_g9  =     length(res9)
Xwins9 = sum(res9==-1)
Draws9 = sum(res9== 0)
Owins9 = sum(res9==+1)
Unfinished9 = sum(res9==+2)
```

```
CombinedFull = c(n_g9,Xwins9/n_g9,Draws9/n_g9,Owins9/n_g9, Unfinished9)

datQ2A = rbind(games5, games8)
kable(datQ2A)
```

Number.of.moves	Number.of.games	X.wins	O.wins	Draws
5	15120	0	1440	0
8	15120	0	1440	0

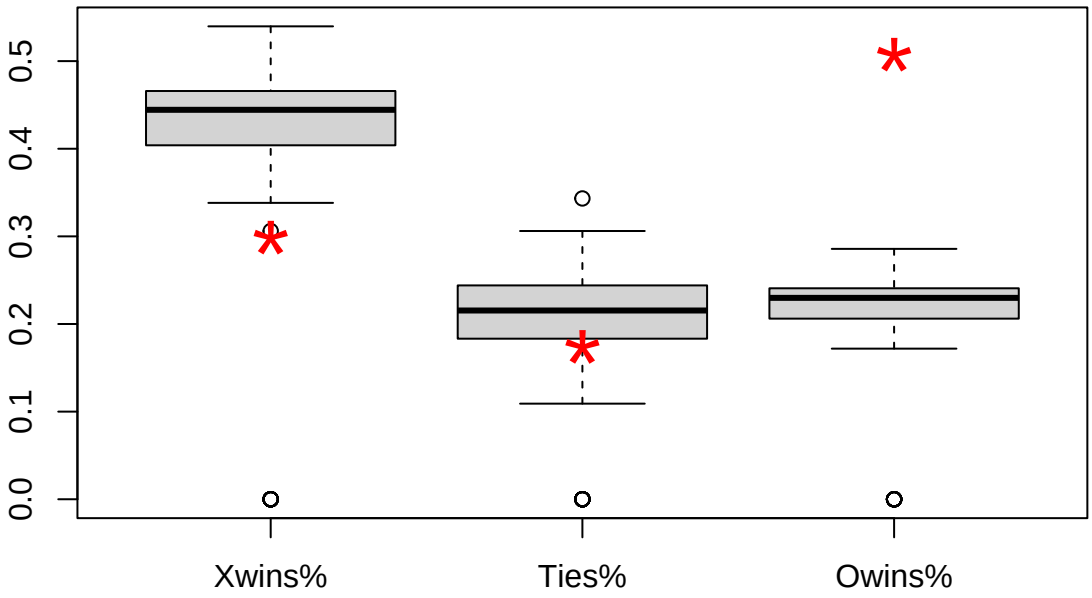
Q2b

```
mcts = function(m, k, alpha)
{
  g = c()
  game_state = rho(m,S)

  randAlpha = runif(1, 0, 1)

  if(game_state$terminal)
  {
    g = c(g,game_state$winner)
    return(g)
  }

  else
  {
    if (randAlpha <= alpha)
    {
      Index = which(m == 0)
      for(i in 1:length(Index))
      {
        x = m
        x[Index[i]]=k
        g = c(g,mcts(x,-1*k, alpha))
      }
      return(g)
    }
    else
    {
      g = c(g,2)
      return(g)
    }

  }
}

m = as.matrix(c(1,1,0,0,0,0,0,0,-1))

resMCTS = mcts(m,-1, 0.95)
n_gMCTS  =  length(resMCTS)
```

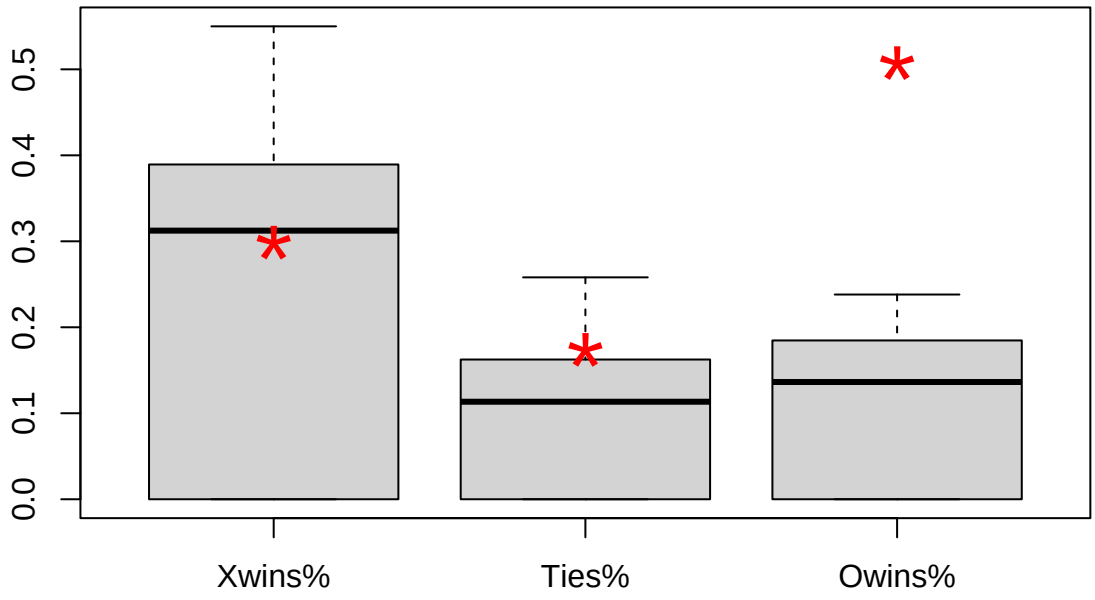
```
XwinsMCTS = sum(resMCTS==-1)
DrawsMCTS = sum(resMCTS== 0)
OwinsMCTS = sum(resMCTS==+1)
UnfinishedMCTS = sum(resMCTS==+2)
MCTS = c(n_gMCTS,XwinsMCTS,DrawsMCTS,OwinsMCTS, UnfinishedMCTS)
```

Q2c

```
m = as.matrix(c(1,1,-1,0,0,0,0,0,-1))

df90 = data.frame()
df70 = data.frame()

for(i in 1:100)
{
  resMCTS90 = mcts(m,1, 0.9)
  resMCTS70 = mcts(m,1, 0.7)

  n_gMCTS90  =  length(resMCTS90)
  XwinsMCTS90 = sum(resMCTS90==-1)
  DrawsMCTS90 = sum(resMCTS90== 0)
  OwinsMCTS90 = sum(resMCTS90==+1)
  UnfinishedMCTS90 = sum(resMCTS==+2)
  Combined90 = c(n_gMCTS90,XwinsMCTS90/n_gMCTS90,DrawsMCTS90/n_gMCTS90,OwinsMCTS90/n_gMCTS90, Unfinishe

  n_gMCTS70  =  length(resMCTS70)
  XwinsMCTS70 = sum(resMCTS70==-1)
  DrawsMCTS70 = sum(resMCTS70== 0)
  OwinsMCTS70 = sum(resMCTS70==+1)
  UnfinishedMCTS70 = sum(resMCTS70==+2)
  Combined70 = c(n_gMCTS70,XwinsMCTS70/n_gMCTS70,DrawsMCTS70/n_gMCTS70,OwinsMCTS70/n_gMCTS70, Unfinishe

  df90 = rbind(df90, Combined90)
  df70 = rbind(df70, Combined70)
}

colnames(df90) = c("Games", "Xwins%", "Ties%", "Owins%", "Incomplete")
colnames(df70) = c("Games", "Xwins%", "Ties%", "Owins%", "Incomplete")

boxplot(df90[, 2:4])
points(c(CombinedFull[2], CombinedFull[3], CombinedFull[4]), pch = "*", col = "red", cex = 3)
```



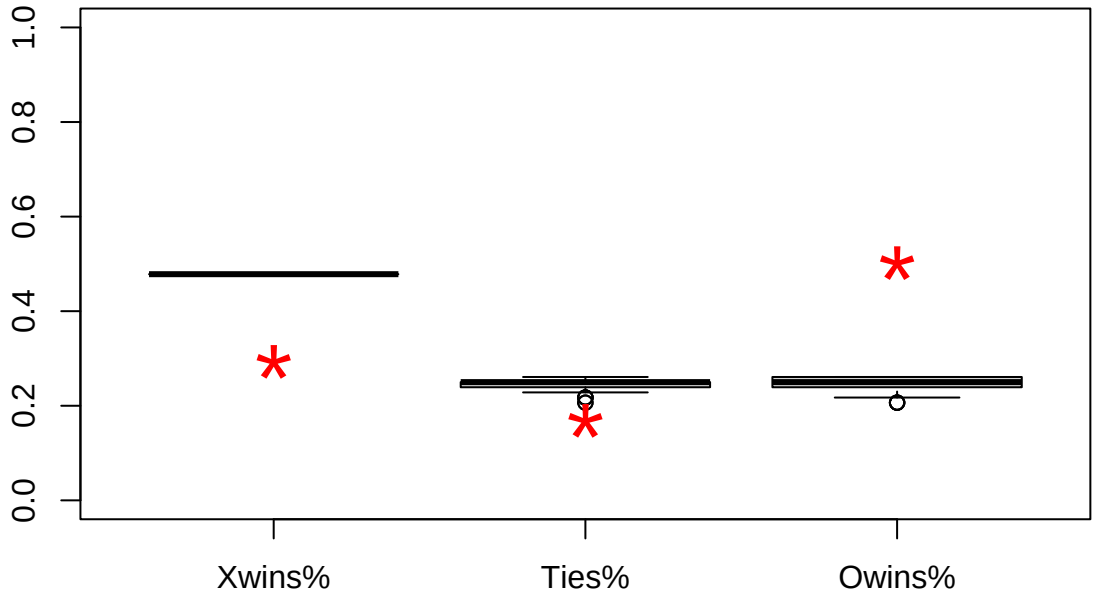

```
boxplot(df70[, 2:4])
points(c(CombinedFull[2], CombinedFull[3], CombinedFull[4]), pch = "*", col = "red", cex = 3)
```



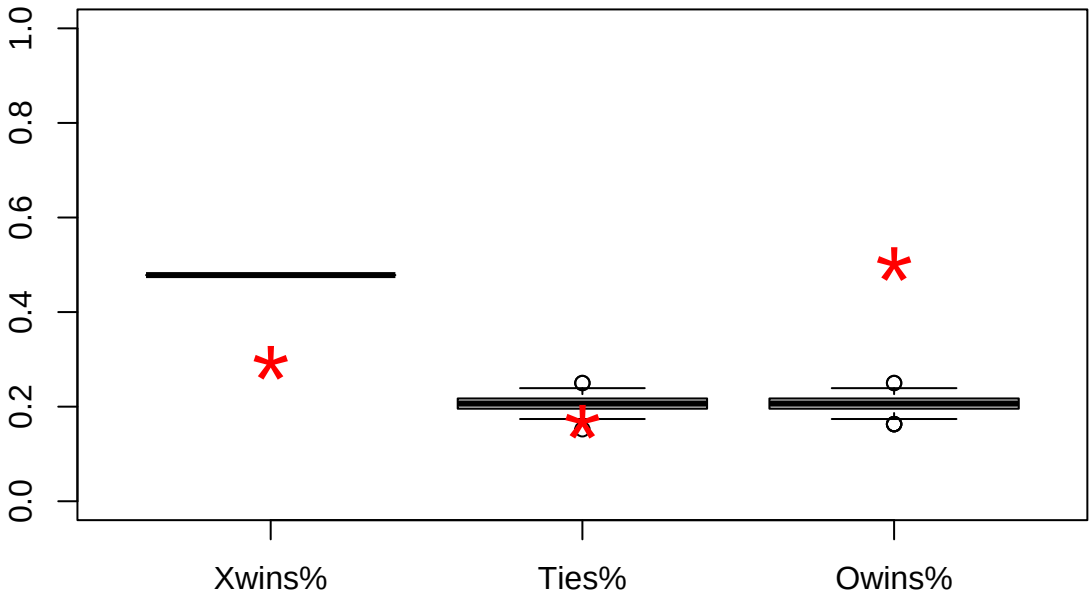

```
mcts_minBranch = function(m, k, alpha, currMoves, minBranch)
{
  g = c()
  game_state = rho(m,S)

  randAlpha = runif(1, 0, 1)

  if(game_state$terminal)
  {
    g = c(g,game_state$winner)
    return(g)
  }

  else
  {
    if (currMoves <= minBranch)
    {
      Index = which(m == 0)
      for(i in 1:length(Index))
      {
        x = m
        x[Index[i]]=k
        g = c(g,mcts_minBranch(x,-1*k, alpha, currMoves + 1, minBranch))
      }
      return(g)
    }
```

```
    else
    {
      if (randAlpha <= alpha)
      {
        Index = which(m == 0)
        for(i in 1:length(Index))
        {
          x = m
          x[Index[i]]=k
          g = c(g,mcts_minBranch(x,-1*k, alpha, currMoves + 1, minBranch))
        }
        return(g)
      }
      else
      {
        g = c(g,2)
        return(g)
      }
    }

  }
}

# m = as.matrix(c(1,1,0,0,0,0,0,0,0,-1))
#
# resMCTS_Branch = mcts_minBranch(m,-1, 0.95, 3, 9)
# n_gMCTS_Branch  =     length(resMCTS_Branch)
# XwinsMCTS_Branch = sum(resMCTS_Branch==-1)
# DrawsMCTS_Branch = sum(resMCTS_Branch== 0)
# OwinsMCTS_Branch = sum(resMCTS_Branch==+1)
# UnfinishedMCTS_Branch = sum(resMCTS_Branch==+2)
# c(n_gMCTS_Branch,XwinsMCTS_Branch,DrawsMCTS_Branch,OwinsMCTS_Branch, UnfinishedMCTS_Branch)

m = as.matrix(c(1,1,-1,0,0,0,0,0,0,-1))

df90_Branch_new = data.frame()
df70_Branch_new = data.frame()

for(i in 1:100)
{
  resMCTS90_new = mcts_minBranch(m,1, 0.9, 4, 7)
  resMCTS70_new = mcts_minBranch(m,1, 0.7, 4, 7)

  n_gMCTS90_new  =  length(resMCTS90_new)
  XwinsMCTS90_new = sum(resMCTS90_new==-1)
  DrawsMCTS90_new = sum(resMCTS90_new== 0)
  OwinsMCTS90_new = sum(resMCTS90_new==+1)
  UnfinishedMCTS90_new = sum(resMCTS90_new==+2)
  Combined90_new = c(n_gMCTS90_new,XwinsMCTS90_new/n_gMCTS90_new,
                     DrawsMCTS90_new/n_gMCTS90_new,OwinsMCTS90_new/n_gMCTS90_new, UnfinishedMCTS90_new)

  n_gMCTS70_new  =  length(resMCTS70_new)
  XwinsMCTS70_new = sum(resMCTS70_new==-1)
```

```
  DrawsMCTS70_new = sum(resMCTS70_new== 0)
  OwinsMCTS70_new = sum(resMCTS70_new==+1)
  UnfinishedMCTS70_new = sum(resMCTS70_new==+2)
  Combined70_new = c(n_gMCTS70_new,XwinsMCTS70_new/n_gMCTS70_new,
                     DrawsMCTS70_new/n_gMCTS70_new,OwinsMCTS70_new/n_gMCTS70_new, UnfinishedMCTS70_new)

  df90_Branch_new = rbind(df90_Branch_new, Combined90_new)
  df70_Branch_new = rbind(df70_Branch_new, Combined70_new)
}

colnames(df90_Branch_new) = c("Games", "Xwins%", "Ties%", "Owins%", "Incomplete")
colnames(df70_Branch_new) = c("Games", "Xwins%", "Ties%", "Owins%", "Incomplete")

boxplot(df90_Branch_new[, 2:4], ylim = c(0, 1))
points(c(CombinedFull[2], CombinedFull[3], CombinedFull[4]), pch = "*", col = "red", cex = 3)
```




```
boxplot(df70_Branch_new[, 2:4], ylim = c(0, 1))
points(c(CombinedFull[2], CombinedFull[3], CombinedFull[4]), pch = "*", col = "red", cex = 3)
```



Q3