Advanced Clinical Trials HW1

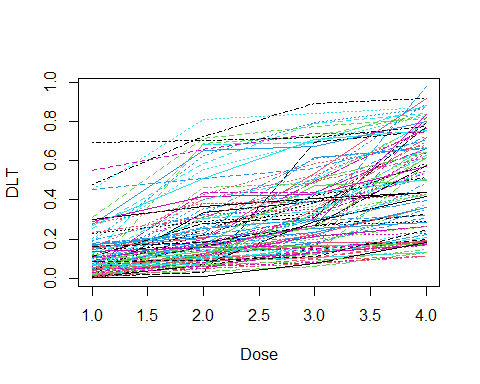
Adam Funk

2023-09-18

## Question 1

The DLT curves are provided below. There does not appear to be any clear pattern in which they fallow other than the monotonically increasing nature built in by design.

#Assignment 1 for Advanced Clinical Trials;  
library(ggplot2)  
#Question 1: Generate 100 dose-toxicity curves using the CLertant and O'Quigley Method  
  
#Set number of doses, number of simulations, and the MTD  
random\_dosetox\_curves <- function(ndose = 4, nsim = 100, mtd = 0.2){  
 #Assign number of doses  
 doses <- ndose  
  
 #Generate Dataframe doses X simulations to export later  
 sims <- data.frame(matrix(nrow = nsim, ncol = doses + 1))  
 mtdfin <- vector("numeric")  
  
 #Create a function that exports one set of possible DLT values  
 sim1 <- function(J = doses){  
 #Identify the mtd for this case  
 mtddose <- floor(runif(1, 1, (doses + 1)))  
 #Sample from a random Beta  
 m <- rbeta(1, max(J - mtddose, 0.5), 1)  
 #Assign the upper bound for DLT probabilities  
 b <- mtd + (1 - mtd) \* m  
 #Generate the random uniform DLTs of doses under that upper bound  
 probs <- sort(runif(J, 0, b))  
 #Idetify the randomly generated MTD  
 found\_mtd <- which.min(abs(probs - mtd))  
 #suppressWarnings(max(max(which((probs - mtd) < 0)), 0))  
 #Assign, Name, and Export the DLTs, Identified MTD, and true MTD dose  
 sim1\_res <- list(probs, found\_mtd, mtddose)  
 names(sim1\_res) <- c("probs", "found\_mtd", "mtddose")  
 return(sim1\_res)  
 }  
  
 #Set Starting value  
 i <- 1  
   
 while(i < nsim + 1){  
 res <- sim1()  
 #Create statement that if the found mtd is equal to the real mtd,  
 #Assign those DLT values to the dataset; else return old dataset  
 if(res$found\_mtd == res$mtddose){  
 sims[i,] <- c(res$probs, res$mtddose)  
 }   
 #Increase value of i only if the DLT values were assigned  
 i <- ifelse(res$found\_mtd == res$mtddose, i + 1, i)  
 }  
 #Return the dataset of all DLT probabilities that matched the criteria  
 return(sims)  
}  
  
  
#Set seed to use throughout  
set.seed(545)  
curves <- random\_dosetox\_curves()  
  
#Plot the DLT results found from the above function;  
matplot(t(curves[, 1:4]), type = "l", xlab = "Dose", ylab = "DLT")



## Question 2

From the design, we properly select the MTD roughly 70% of the time in the initial simulated study. This is a very good PCS. The boxplot shows that although most curves perform well, they are not as well performing as the first model, with an average of 57.3%, and a median and range of 55.9% (1.5%, 100.0%).

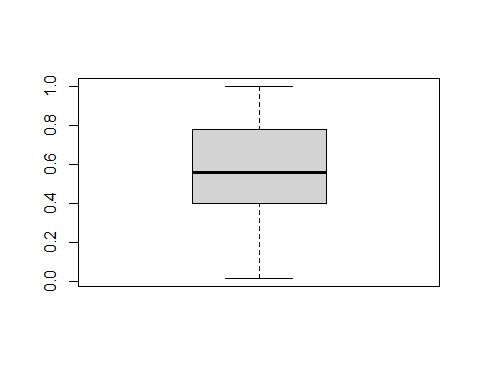
#Question 2: Nonparametric Benchmark  
non\_bench <- function(theta = 0.3, npat = 24, nsim = 5000, dltcuts = c(0.05, 0.15, 0.3, 0.5)){  
 dltval <- vector("numeric")  
 for(j in 1:nsim){  
 #Set random uniform for a set of patients  
 randlts <- runif(npat)  
 #Set empty vectors  
 simdlt <- vector("numeric")  
 #Identify the minimum for each simulation  
 for(i in 1:length(dltcuts)){  
 simdlt[i] <- sum(randlts < dltcuts[i]) / npat  
 }  
 #Identify MTD  
 dltval[j] <- which.min(abs(simdlt - theta))  
 }  
 #Return the MTD value  
 return(dltval)  
}  
  
#Set Value to a table  
table2 <- non\_bench()  
  
#Get number of times each dose is selected as mtd  
doseselect <- c(sum(table2 == 1), sum(table2 == 2), sum(table2 == 3), sum(table2 == 4))/ (5000)  
doseselect

## [1] 0.0008 0.1708 0.6914 0.1370

#Get Accuracy Index  
1 - 4 \* (abs(0.05 - 0.3) \* doseselect[1] + abs(0.15 - 0.3) \* doseselect[2] + abs(0.3 - 0.3) \* doseselect[3] + abs(0.5 - 0.3) \* doseselect[4]) /  
 (abs(0.05 - 0.3) + abs(0.15 - 0.3) + abs(0.3 - 0.3) + abs(0.5 - 0.3))

## [1] 0.6452

#Set an empty matrix   
vals <- matrix(nrow = 100, ncol = 5)  
#Set i  
i <- 1  
#Replace the empty matrix with PCS values for each curve  
for(i in 1:100){  
   
 table2.2 <- non\_bench(theta = 0.2, npat = 24, nsim = 1000, dltcuts = as.vector(curves[i, 1:4]))  
 doseselect <- c(sum(table2.2 == 1), sum(table2.2 == 2), sum(table2.2 == 3), sum(table2.2 == 4)) / (1000)  
 vals[i, ] <- c(doseselect, curves[i, 5])  
   
}  
  
  
#Isolate the true MTD values found  
mtdvals <- vals[,5]  
#Match those MTD values to the appropriate column  
matchvals <- matrix(nrow = 100, ncol = 2)  
for(i in 1:100){  
 matchvals[i, 1] <- vals[i, mtdvals[i]]  
 matchvals[i, 2] <- mtdvals[i]  
}  
  
#Plot the boxplot of the PCS values  
boxplot(matchvals[, 1])



#Summary statistics of PCS values  
mean(matchvals[, 1])

## [1] 0.57285

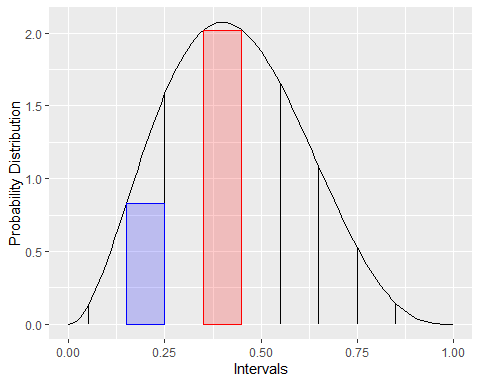
quantile(matchvals[, 1])

## 0% 25% 50% 75% 100%   
## 0.01500 0.39975 0.55900 0.77700 1.00000

## Question 3

Below is the posterior curve with the red shaded region the current and the blue the target dlt rates. From these shaded areas, we would recommend to deescalate the dosage. However, this is not possible, but the dosage is not yet eliminated by the stopping rules. Thus we do not yet stop the trial despite there being no dose to deescalate to.

#Question 3: Keyboard design  
#Set Beta (1,1)  
a <- 1  
b <- 1  
#Target DLT  
theta <- 0.2  
#Epsilons of intervals  
ep <- 0.05  
#Number of DLTs  
succ <- 2  
#Number of non-DLTs  
fail <- 3  
#Get the length of each key  
keyleng <- 2 \* ep  
  
keyboard <- function(a =1, b = 1, theta = 0.2, ep = 0.05, succ = 2, fail = 3){  
#Generate proper sequence of   
intervals <- seq(from = ((theta - ep) %% keyleng), to = 1, by = keyleng)  
#Set empty vector to populate values, their names, and denote the true interval  
val <- matrix(nrow = 1, ncol = length(intervals) - 1)  
valnames <- matrix(nrow = 1, ncol = length(intervals) - 1)  
trueint <- matrix(nrow = 1, ncol = length(intervals) - 1)  
  
#Generate the probabilities from the beta distribution given our data  
for(i in 1:(length(intervals) - 1)){  
 val[i] <- pbeta(intervals[i + 1], a + succ, b + fail) - pbeta(intervals[i], a + succ, b + fail)  
 valnames[i] <- c(paste0(intervals[i], ", ", intervals[i + 1]))  
 trueint[i] <- (intervals[i] < theta & intervals[i + 1] > theta)  
}  
  
#Identify the target interval  
targetint <- as.numeric(which(trueint == T))  
colors <- c("Current" = "red", "Target" = "red")  
  
#Plot the distribution  
keyplot <- qplot(seq(0, 1, 0.01), geom = "blank") +   
 stat\_function(aes(1), fun = dbeta, colour = "black", n = 101,   
 args = list(a + succ, b + fail)) +  
 geom\_segment(aes(x = intervals, y = 0, xend = intervals, yend = dbeta(intervals, a + succ, b + fail))) +   
 #Highlight the highest probability area in red  
 annotate('rect', xmin = intervals[which.max(val)], xmax = intervals[which.max(val) + 1],  
 ymin = 0, ymax = dbeta(seq(intervals[which.max(val)], intervals[which.max(val)], 0.01), a + succ, b + fail),  
 color = "red", alpha = 0.2, fill = "red") +  
 #Highligh the target probability area in blue  
 annotate('rect', xmin = intervals[targetint], xmax = intervals[targetint + 1],  
 ymin = 0, ymax = dbeta(seq(intervals[targetint], intervals[targetint], 0.01), a + succ, b + fail),  
 color = "blue", alpha = 0.2, fill = "blue") +   
 labs(x = "Intervals",  
 y = "Probability Distribution",  
 color = "Legend")  
 scale\_color\_manual(values = colors)  
  
return(keyplot)  
  
}  
  
#Give results desired in assignment  
keyboard()



## Question 4

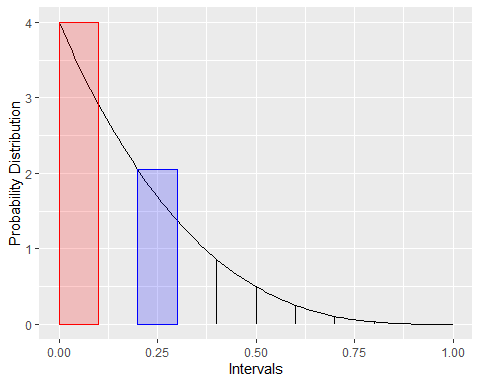
#Question 4: Compare 3 + 3, BOIN, and Keyboard  
library(UBCRM)  
library(BOIN)  
set.seed(545)  
latent\_tox <- runif(24)  
round(latent\_tox, 3)

## [1] 0.681 0.905 0.692 0.442 0.218 0.704 0.137 0.464 0.253 0.754 0.033 0.839  
## [13] 0.751 0.646 0.569 0.712 0.488 0.618 0.796 0.179 0.467 0.930 0.018 0.441

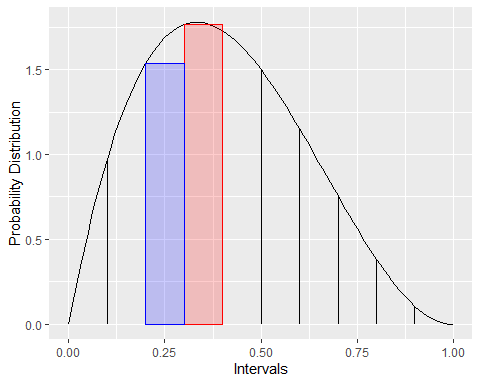
dlts <- c(0.05, 0.13, 0.25, 0.38)  
  
#3 plus 3 design  
set.seed(545)  
s3p3 <- sim3p3(dlts, seed = 545)  
  
#BOIN design  
set.seed(545)  
boinres <- get.oc(target = 0.25, p.true = dlts, ncohort = 8, cohortsize = 3, ntrial = 1, see = 545)  
get.boundary(target = 0.25, ncohort = 8, cohortsize = 3)

## $lambda\_e  
## [1] 0.1968009  
##   
## $lambda\_d  
## [1] 0.2983922  
##   
## $boundary\_tab  
##   
## Number of patients treated 3 6 9 12 15 18 21 24  
## Escalate if # of DLT <= 0 1 1 2 2 3 4 4  
## Deescalate if # of DLT >= 1 2 3 4 5 6 7 8  
## Eliminate if # of DLT >= 3 4 5 6 7 8 9 10  
##   
## $full\_boundary\_tab  
##   
## Number of patients treated 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  
## Escalate if # of DLT <= 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3  
## Deescalate if # of DLT >= 1 1 1 2 2 2 3 3 3 3 4 4 4 5 5 5 6 6 6 6  
## Eliminate if # of DLT >= NA NA 3 3 3 4 4 4 5 5 6 6 6 7 7 7 8 8 8 9  
##   
## Number of patients treated 21 22 23 24  
## Escalate if # of DLT <= 4 4 4 4  
## Deescalate if # of DLT >= 7 7 7 8  
## Eliminate if # of DLT >= 9 9 10 10  
##   
## attr(,"class")  
## [1] "boin"

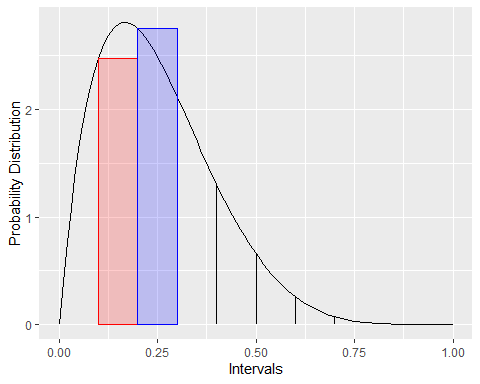
#Keyboard  
#Cohort 1 & 2 & 6  
keyboard(a = 1, b = 1, theta = 0.25, ep = 0.05, succ = 0, fail = 3)



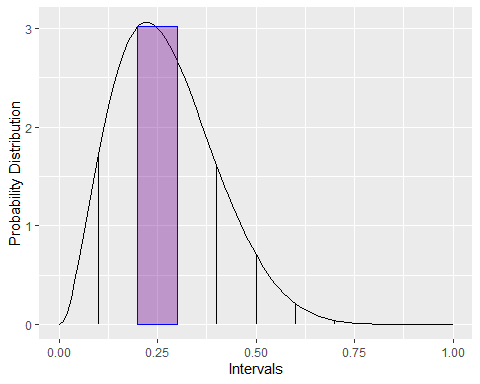
#Cohort 3  
keyboard(a = 1, b = 1, theta = 0.25, ep = 0.05, succ = 1, fail = 2)



#Cohort 4 & 5 & 7  
keyboard(a = 1, b = 1, theta = 0.25, ep = 0.05, succ = 1, fail = 5)



#Final Allocation  
keyboard(a = 1, b = 1, theta = 0.25, ep = 0.05, succ = 2, fail = 7)



## Question 5

The i3 + 3 design gives a PCS of approximately 54%, which is lower than the benchmark value of 70%, indicating as expected it performs worse. This trend carries over to the generate curves where we have a PCS mean of 46.1% with a median and range of 41.9% (0.4%, 1.00%). The boxplots look similar, just shifted down a little as a result of the worse performance of this method.

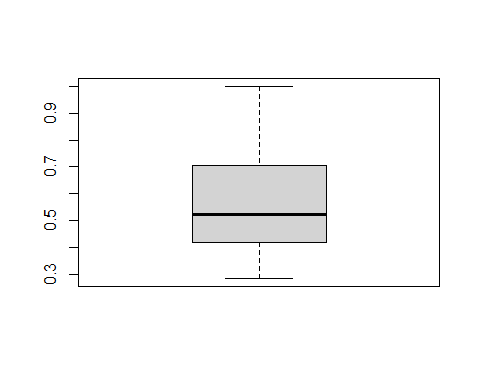
#Question 5: i3 + 3 design  
  
#Create i3 + 3 function  
i33 <- function(mtd = 0.3, ep = 0.05, n = 3, ncohort = 8, dltprobs = c(0.05, 0.15, 0.3, 0.5)){  
 #Set upper and lower bounds of the interval  
 lbmtd <- mtd - ep  
 upmtd <- mtd + ep  
 #Set initial dose level  
 dose <- 1  
 #Create empty vectors to fill in  
 dltall <- vector("numeric")  
 dosevec <- vector("numeric")  
 for(i in 1:ncohort){  
 #Generate number of random values  
 x <- runif(n)  
 #Decide if those random values are greater than the DLT cutoff  
 dlt <- (x < dltprobs[dose])  
 #Get a vector of the dose level  
 dosevals <- rep(dose, n)  
 #Add new values to vectors  
 dltall <- c(dltall, dlt)  
 dosevec <- c(dosevec, dosevals)  
 #Set value of dlt over mtd  
 dltmtdrat <- sum(dltall[dosevec == dose])/length(dltall[dosevec == dose])  
 #Set value of dlt - 1 over mtd  
 dltmtdratm1 <- (sum(dltall[dosevec == dose]) - 1)/length(dltall[dosevec == dose])  
 #Create conditional statement as to what the next dose level should be  
 if(dltmtdrat < lbmtd){  
 dose <- min(dose + 1, length(dltprobs))  
 } else if (lbmtd <= dltmtdrat & dltmtdrat <= upmtd){  
 dose <- dose  
 } else if (upmtd < dltmtdrat & dltmtdratm1 < lbmtd){  
 dose <- dose  
 } else {dose <- max(dose - 1, 1)}  
   
 }  
 #Return dataframe as output  
 return(as.data.frame(cbind(dltall, dosevec, dose)))  
}  
  
set.seed(545)  
#Get final dosages  
table5 <- replicate(5000, i33()$dose)  
#Identify how often we are choosing those doses  
doseselect <- c(sum(table5 == 1), sum(table5 == 2), sum(table5 == 3), sum(table5 == 4))/ (24 \* 5000)  
doseselect

## [1] 0.0100 0.1954 0.5432 0.2514

#Set an empty matrix   
vals <- matrix(nrow = 100, ncol = 5)  
  
#Set i  
i <- 1  
#Use the curves to get PCS estimates;  
for(i in 1:100){  
 table5.2 <- replicate(1000, i33(mtd = 0.2, n = 3, ncohort = 8, dltprobs = as.vector(curves[i, 1:4]))$dose)  
 doseselect <- c(sum(table5.2 == 1), sum(table5.2 == 2), sum(table5.2 == 3), sum(table5.2 == 4))/ (24 \* 1000)  
 vals[i, 1:5] <- c(doseselect, curves[i, 5])  
}  
  
vals

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.400 0.499 0.081 0.020 2  
## [2,] 0.041 0.083 0.245 0.631 4  
## [3,] 0.379 0.501 0.105 0.015 2  
## [4,] 0.422 0.519 0.057 0.002 2  
## [5,] 0.741 0.259 0.000 0.000 1  
## [6,] 0.118 0.246 0.253 0.383 4  
## [7,] 0.043 0.191 0.283 0.483 4  
## [8,] 0.106 0.349 0.455 0.090 2  
## [9,] 0.557 0.429 0.013 0.001 1  
## [10,] 0.026 0.146 0.303 0.525 4  
## [11,] 0.014 0.078 0.184 0.724 4  
## [12,] 0.998 0.002 0.000 0.000 1  
## [13,] 0.254 0.285 0.308 0.153 2  
## [14,] 0.752 0.198 0.046 0.004 1  
## [15,] 0.001 0.010 0.127 0.862 4  
## [16,] 0.413 0.287 0.191 0.109 2  
## [17,] 0.020 0.134 0.313 0.533 4  
## [18,] 0.460 0.428 0.096 0.016 1  
## [19,] 0.259 0.553 0.177 0.011 2  
## [20,] 0.065 0.186 0.274 0.475 4  
## [21,] 0.138 0.369 0.414 0.079 2  
## [22,] 0.317 0.310 0.283 0.090 2  
## [23,] 1.000 0.000 0.000 0.000 1  
## [24,] 0.005 0.033 0.141 0.821 4  
## [25,] 0.000 0.300 0.699 0.001 2  
## [26,] 0.029 0.212 0.306 0.453 4  
## [27,] 0.736 0.264 0.000 0.000 1  
## [28,] 0.816 0.182 0.002 0.000 1  
## [29,] 0.364 0.514 0.120 0.002 2  
## [30,] 0.202 0.559 0.236 0.003 2  
## [31,] 0.010 0.318 0.552 0.120 3  
## [32,] 0.003 0.044 0.116 0.837 4  
## [33,] 0.017 0.086 0.307 0.590 4  
## [34,] 0.104 0.244 0.386 0.266 3  
## [35,] 0.357 0.481 0.139 0.023 2  
## [36,] 0.043 0.231 0.426 0.300 3  
## [37,] 0.008 0.085 0.325 0.582 4  
## [38,] 0.543 0.443 0.014 0.000 1  
## [39,] 0.022 0.128 0.295 0.555 4  
## [40,] 0.425 0.296 0.256 0.023 2  
## [41,] 0.909 0.087 0.004 0.000 1  
## [42,] 0.154 0.197 0.226 0.423 4  
## [43,] 0.689 0.195 0.108 0.008 1  
## [44,] 0.422 0.398 0.169 0.011 2  
## [45,] 0.009 0.078 0.634 0.279 3  
## [46,] 0.940 0.060 0.000 0.000 1  
## [47,] 0.391 0.391 0.169 0.049 2  
## [48,] 0.073 0.262 0.347 0.318 3  
## [49,] 0.995 0.005 0.000 0.000 1  
## [50,] 0.015 0.132 0.275 0.578 4  
## [51,] 0.135 0.550 0.296 0.019 2  
## [52,] 0.092 0.326 0.309 0.273 3  
## [53,] 0.823 0.174 0.003 0.000 1  
## [54,] 0.185 0.417 0.348 0.050 2  
## [55,] 0.394 0.426 0.165 0.015 2  
## [56,] 0.045 0.307 0.439 0.209 3  
## [57,] 0.960 0.040 0.000 0.000 1  
## [58,] 0.457 0.482 0.055 0.006 1  
## [59,] 0.084 0.399 0.466 0.051 2  
## [60,] 0.524 0.448 0.027 0.001 1  
## [61,] 0.886 0.088 0.021 0.005 1  
## [62,] 0.007 0.113 0.282 0.598 4  
## [63,] 0.145 0.272 0.312 0.271 3  
## [64,] 0.653 0.295 0.052 0.000 1  
## [65,] 0.108 0.141 0.258 0.493 4  
## [66,] 0.895 0.089 0.013 0.003 1  
## [67,] 0.101 0.388 0.366 0.145 2  
## [68,] 0.845 0.127 0.026 0.002 1  
## [69,] 0.182 0.199 0.370 0.249 3  
## [70,] 0.487 0.354 0.157 0.002 1  
## [71,] 0.042 0.500 0.435 0.023 2  
## [72,] 0.004 0.113 0.720 0.163 3  
## [73,] 0.216 0.531 0.231 0.022 2  
## [74,] 0.002 0.027 0.113 0.858 4  
## [75,] 0.047 0.472 0.471 0.010 2  
## [76,] 0.134 0.384 0.323 0.159 2  
## [77,] 0.890 0.109 0.001 0.000 1  
## [78,] 0.189 0.429 0.344 0.038 2  
## [79,] 1.000 0.000 0.000 0.000 1  
## [80,] 0.046 0.470 0.447 0.037 2  
## [81,] 0.058 0.203 0.299 0.440 4  
## [82,] 0.994 0.005 0.001 0.000 1  
## [83,] 0.115 0.188 0.294 0.403 4  
## [84,] 0.006 0.085 0.277 0.632 4  
## [85,] 0.335 0.297 0.312 0.056 2  
## [86,] 0.380 0.426 0.191 0.003 2  
## [87,] 0.056 0.419 0.472 0.053 2  
## [88,] 0.334 0.545 0.121 0.000 2  
## [89,] 0.544 0.421 0.034 0.001 1  
## [90,] 0.016 0.093 0.247 0.644 4  
## [91,] 0.000 0.010 0.208 0.782 4  
## [92,] 0.020 0.086 0.253 0.641 4  
## [93,] 0.015 0.407 0.501 0.077 2  
## [94,] 0.202 0.308 0.362 0.128 3  
## [95,] 0.025 0.359 0.573 0.043 3  
## [96,] 0.232 0.326 0.407 0.035 2  
## [97,] 0.052 0.104 0.358 0.486 4  
## [98,] 0.634 0.333 0.033 0.000 1  
## [99,] 0.008 0.052 0.180 0.760 4  
## [100,] 0.382 0.356 0.204 0.058 2

#Isolate the true MTD values found  
mtdvals <- vals[,5]  
#Match those MTD values to the appropriate column  
matchvals <- matrix(nrow = 100, ncol = 2)  
for(i in 1:100){  
 matchvals[i, 1] <- vals[i, mtdvals[i]]  
 matchvals[i, 2] <- mtdvals[i]  
}  
  
#Plot the boxplot of the PCS values  
boxplot(matchvals[, 1], subset = matchvals[, 2])



#Summary statistics of PCS values  
mean(matchvals[, 1])

## [1] 0.56703

quantile(matchvals[, 1])

## 0% 25% 50% 75% 100%   
## 0.28500 0.41850 0.52450 0.69675 1.00000