BIOS 6301 Assignment 4.0

James Lopez

11/2/2016

Grade 33/50

Comments: Hi James, thanks for your notes on slack, and I hope your interviews went well! To be fair, I did take off 5 for the assignment being turned in late. Don't despair though, this is just one assignment though and you can still be in good shape for the class. Also for future reference, you can probably get an extension from Cole, just so long as you get it okayed before the due date.

In the football problem you've added a requisite that there is at least 1 player in each position via the following code: x2[qb.ix, 'marg'] <- x2[qb.ix,'points'] - x2[qb.ix[max(1,posReq['qb'])*nTeams],'points'] . I did not take off for this because it was a logical assumption. But FYI, Cole's intent is to allow for zero players in a position for the fantasy team.

I'd be happy to sit down during office hours and review the problems you missed on.

Jonathan

Question 1

15 points

```
secant <- function(f, x0, x1, tol = 1e-7, maxiter = 1000) {
  for (i in 1:maxiter) {
    x2 <- x1 - f(x1) / ((f(x1) - f(x0)) / (x1 - x0))
    if (abs(x2 - x1) < tol) {
      return(x2)
    }
    x0 <- x1
    x1 <- x2
  }
}

funct <- function(x) cos(x) - x
secant(funct, 0, 1)
```

```
## [1] 0.7390851
```

```
system.time(replicate(1000, secant(funct, 0, 1)))
```

```
##    user  system elapsed
##   0.032   0.004   0.036
```

```
newton <- function(guess, f, fp, tol=10e-7, maxiter=1000) {
  i <- 1
  while(abs(f(guess))>tol && i < maxiter) {
    guess <- guess - f(guess)/fp(guess)
    i <- i+1
  }
  if(i == maxiter) {
    return(NULL)
```

```
  }
  else {
  }
  guess
}

f <- function(x) -sin(x)-1
fp <- function(x) -cos(x)
newton(10, f, fp)
```

```
## [1] 10.99471
```

```
system.time(replicate(1000, newton(10,f,fp)))
```

```
##    user  system elapsed
##   0.046   0.004   0.050
```

The secant method takes MUCH less time – 0.16 seconds faster than the Newton-Raphson method.

JC Grading -3 Check the newton rhapson function. It converges to 10.99471 rathern than 0.7390851.

2 The game of craps is played as follows. First, you roll two six-sided dice; let x be the sum of the dice on the first roll. If x = 7 or 11 you win, otherwise you keep rolling until either you get x again, in which case you also win, or until you get a 7 or 11, in which case you lose.

18 points

part 1) The instructor should be able to easily import and run your program (function), and obtain output that clearly shows how the game progressed. Set the RNG seed with set.seed(100) and show the output of three games.

```
craps <- function(n) {
  win<-0
  loss<-0
  groll<-0
  results = data.frame(win, loss)
  set.seed(100)
  for (y in 1:n)  {
    a <- sum(ceiling(6*runif(2)))
    numroll <- 1
    if  ((a==7)||(a==11)) {
      results[,'win'] <- results[,'win'] + 1
      print("First roll win!")
      print(numroll)
    }
```

```
    z<-(a == 7)||(a == 11)
    if  (z==FALSE)  {
      point <- a
      repeat  {
        a2 <- sum(ceiling(6*runif(2)))
        numroll <- numroll + 1
        if  ((a2 == 7) || (a2 == 11)) {
          results[,'loss'] <- results[,'loss'] + 1
          print("You lost!")
          print(numroll)
          break
        }
        if  ((a2==point)) {
          results[,'win'] <- results[,'win'] + 1
          print("You won!")
          print(numroll)
          break
        }
      }
    }
    groll <- groll + numroll

    results[,'rolls'] <- groll
  }
  print(results)
}

craps(3)
```

```
## [1] "You lost!"
## [1] 14
## [1] "You lost!"
## [1] 4
## [1] "You lost!"
## [1] 2
##   win loss rolls
## 1   0    3    20
```

In three games, I had 3 losses over a total of 20 rolls. The printed output messages tells how many rolls it took to reach each result (e.g. 4 rolls for the second loss and 2 rolls for the third loss) "First roll win!" message means you rolled a 7 or 11 on the first roll.

2 part 2 Find a seed that will win ten straight games. Consider adding an argument to your function that disables output. Show the output of the ten games.

```
result <- logical(1)
craps <- function(n, s) {
  win<-0
  loss<-0
  groll<-0
```

```
  results = data.frame(win, loss)
  set.seed(s)
  for (y in 1:n)  {
    a <- sum(ceiling(6*runif(2)))
    numroll <- 1
    if  ((a==7)||(a==11)) {
      results[,'win'] <- results[,'win'] + 1
      result <- T
      print("First roll win!")
      print(numroll)
    }
    z<-(a == 7)||(a == 11)
    if  (z==FALSE)  {
      point <- a
      repeat  {
        a2 <- sum(ceiling(6*runif(2)))
        numroll <- numroll + 1
        if  ((a2 == 7) || (a2 == 11)) {
          results[,'loss'] <- results[,'loss'] + 1
          result <- F
          print("You lost!")
          print(numroll)
          break
        }
        if  ((a2==point)) {
          results[,'win'] <- results[,'win'] + 1
          result <- T
          print("You won!")
          print(numroll)
          break
        }
      }
    }
    groll <- groll + numroll

    results[,'rolls'] <- groll
  }
  print(results)
}
```

```
s1 <- 1
while(sum(replicate(10, result) == T) != 10) {
  s1 <- s1+1
}
```

The seed was 880.

JC Grading -7

This code creates an infinite. Look at the value of result, which is 1.

```
craps(10, s1)
```

3 Obtain a copy of the football-values lecture. Save the five 2016 CSV files in your working directory.

Modify the code to create a function. This function will create dollar values given information (as arguments) about a league setup. It will return a data.frame and write this data.frame to a CSV file. The final data.frame should contain the columns 'PlayerName', 'pos', 'points', 'value' and be orderd by value descendingly. Do not round dollar values.

Note that the returned data.frame should have sum(posReq)*nTeams rows.

Define the function as such:

```
ffvalues <- function(path, file='outfile.csv', nTeams=12, cap=200, posReq=c(qb=1, rb=2, wr=3, te=1, k=1
                     points=c(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
                           rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)) {
  setwd(path)
  k  <- read.csv("proj_k16.csv")
  qb <- read.csv("proj_qb16.csv")
  rb <- read.csv("proj_rb16.csv")
  te <- read.csv("proj_te16.csv")
  wr <- read.csv("proj_wr16.csv")

  k[,'pos']  <- 'k'
  qb[,'pos'] <- 'qb'
  rb[,'pos'] <- 'rb'
  te[,'pos'] <- 'te'
  wr[,'pos'] <- 'wr'
  cols <- unique(c(names(k), names(qb), names(rb), names(te), names(wr)))

  k[,setdiff(cols, names(k))]   <- 0
  qb[,setdiff(cols, names(qb))] <- 0
  rb[,setdiff(cols, names(rb))] <- 0
  te[,setdiff(cols, names(te))] <- 0
  wr[,setdiff(cols, names(wr))] <- 0
  x <- rbind(k[,cols], qb[,cols], rb[,cols], te[,cols], wr[,cols])
  x[,'p_fg']        <- x[,'fg']*points[1]
  x[,'p_xpt']       <- x[,'xpt']*points[2]
  x[,'p_pass_yds']  <- x[,'pass_yds']*points[3]
  x[,'p_pass_tds']  <- x[,'pass_tds']*points[4]
  x[,'p_pass_ints'] <- x[,'pass_ints']*points[5]
  x[,'p_rush_yds']  <- x[,'rush_yds']*points[6]
  x[,'p_rush_tds']  <- x[,'rush_tds']*points[7]
  x[,'p_fumbles']   <- x[,'fumbles']*points[8]
  x[,'p_rec_yds']   <- x[,'rec_yds']*points[9]
```

```
  x[,'p_rec_tds']   <- x[,'rec_tds']*points[10]
  x[,'points'] <- rowSums(x[,grep("^p_", names(x))])
  x2 <- x[order(x[,'points'], decreasing=TRUE),]
  k.ix  <- which(x2[,'pos']=='k')
  qb.ix <- which(x2[,'pos']=='qb')
  rb.ix <- which(x2[,'pos']=='rb')
  te.ix <- which(x2[,'pos']=='te')
  wr.ix <- which(x2[,'pos']=='wr')
  x2[qb.ix, 'marg'] <- x2[qb.ix,'points'] - x2[qb.ix[max(1,posReq['qb'])*nTeams],'points']
  x2[rb.ix, 'marg'] <- x2[rb.ix,'points'] - x2[rb.ix[max(1,posReq['rb'])*nTeams],'points']
  x2[wr.ix, 'marg'] <- x2[wr.ix,'points'] - x2[wr.ix[max(1,posReq['wr'])*nTeams],'points']
  x2[te.ix, 'marg'] <- x2[te.ix,'points'] - x2[te.ix[max(1,posReq['te'])*nTeams],'points']
  x2[k.ix,  'marg'] <- x2[k.ix, 'points'] -  x2[k.ix[max(1,posReq['k'])*nTeams],  'points']
  x3 <- x2[x2[,'marg'] >= 0,]
  x3 <- x3[order(x3[,'marg'], decreasing=TRUE),]
  rownames(x3) <- NULL
  x3[,'value'] <- x3[,'marg']*(nTeams*cap-nrow(x3))/sum(x3[,'marg']) + 1
  x4 <- x3[,c('PlayerName','pos','points','marg','value')]
  x4[,'marg'] <- NULL
  write.csv(x4, file)
  return(x4)

}
```

part 1 i) How many players are worth more than $20?

```
f1 <- ffvalues('.')
length(which(f1[,'value'] > 20))
```

```
## [1] 46
```

There are 46 players worth more than $20

part 1 ii) Who is 15th most valuable running back (rb)?

```
f1[which(f1[,'pos'] == 'rb')[15],]
```

```
##     PlayerName pos points    value
## 47 Carlos Hyde  rb 145.47 19.76574
```

Carlos Hyde

part 2 i) How many players are worth more than $20?

```
f2 <- ffvalues(getwd(), '16team.csv', nTeams=16, cap=150)
length(which(f2[,'value'] > 20))
```

```
## [1] 49
```

There are 49 players worth more than $20

part 2 ii) How many wide receivers (wr) are in the top 40?

```
length(which(f2[1:40, 'pos'] == 'wr'))
```

```
## [1] 18
```

There are 18 wide receivers in the top 40.

part 3 i) How many players are worth more than $20?

```
f3 <- ffvalues('.', 'qbheavy.csv', posReq=c(qb=2, rb=2, wr=3, te=1, k=0),
               points=c(fg=0, xpt=0, pass_yds=1/25, pass_tds=6, pass_ints=-2,
                        rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6))
length(which(f3[,'value'] > 20))
```

```
## [1] 50
```

There are 50 players worth more than $20

part 3 ii) How many quarterbacks (qb) are in the top 30?

```
length(which(f3[1:30, 'pos'] == 'qb'))
```

```
## [1] 10
```

There are 10 qb's in the top 30.

4

part 1 Which function has the most arguments?

```
library(plyr)
objs <- mget(ls("package:base"), inherits = TRUE)
funs <- Filter(is.function, objs)

argument.nums <- laply(funs, function(x)(length(formals(x))))
max.args <- which(argument.nums == max(argument.nums))
names(funs[max.args])
```

```
## [1] "scan"
```

part 2 How many functions have no arguments?

```
length(which(arg_length == 0))
```

225 functions

JC Grading - 2

Creates an error: Error in which(arg_length == 0) : object ‘arg_length’ not found