Bios 6301: Final Project

Lingjun Fu

12/14/2015

Due Monday, 14 December, 6:00 PM

200 points total.

Submit a single knitr file (named `final.rmd`), along with a valid PDF output file. Add your name as `author` to the file's metadata section. Raw R code/output or word processor files are not acceptable.

All work should be done by the student, please no collaboration. You may ask the instructor for help or clarification.

Obtain a copy of the football-values lecture – make sure to update this repository if you have previously cloned it. Save the six 2015 CSV files in your working directory (note the new file `nfl_current15.csv`). You may utilize assignment 4, question 3 in your solution.

Task 1: Finding Residuals (80 points)

At the beginning of the course we examined projections for the 2015 NFL season. With the season ~60% completed, let's compare the observed values to the estimated values. Place all code at the end of the instructions.

1. Read and combine the projection data (five files) into one data set, adding a position column.

2. The NFL season is 17 weeks long, and 10 weeks have been completed. Each team plays 16 games and has one week off, called the bye week. Four teams have yet to have their bye week: CLE, NO, NYG, PIT. These four teams have played ten games, and every other team has played nine games. Multiply the numeric columns in the projection data by the percentage of games played (for example, 10/16 if team is PIT).

3. Sort and order the data by the `fpts` column descendingly. Subset the data by keeping the top 20 kickers, top 20 quarterbacks, top 40 running backs, top 60 wide recievers, and top 20 tight ends. Thus the projection data should only have 160 rows.

4. Read in the observed data (`nfl_current15.csv`)

5. Merge the projected data with the observed data by the player's name. Keep all 160 rows from the projection data. If observed data is missing, set it to zero.

   You can directly compare the projected and observed data for each player. There are fifteen columns of interest:

```
##                      Name projected_col observed_col
## 1             field goals            fg          FGM
## 2   field goals attempted           fga          FGA
## 3            extra points           xpt          XPM
## 4       passing attempts       pass_att     Att.pass
## 5    passing completions       pass_cmp     Cmp.pass
## 6          passing yards       pass_yds     Yds.pass
## 7     passing touchdowns       pass_tds      TD.pass
## 8  passing interceptions      pass_ints     Int.pass
## 9       rushing attempts       rush_att     Att.rush
```

```
## 10          rushing yards      rush_yds     Yds.rush
## 11     rushing touchdowns      rush_tds      TD.rush
## 12     receiving attempts       rec_att    Rec.catch
## 13        receiving yards       rec_yds    Yds.catch
## 14   receiving touchdowns       rec_tds     TD.catch
## 15                fumbles       fumbles          Fmb
```

6. Take the difference between the observed data and the projected data for each category. Split the data by position, and keep the columns of interest.

You will now have a list with five elements. Each element will be a matrix or data.frame with 15 columns.

```r
### step 1
k <- read.csv('proj_k15.csv', header=TRUE, stringsAsFactors=FALSE)
qb <- read.csv('proj_qb15.csv', header=TRUE, stringsAsFactors=FALSE)
rb <- read.csv('proj_rb15.csv', header=TRUE, stringsAsFactors=FALSE)
te <- read.csv('proj_te15.csv', header=TRUE, stringsAsFactors=FALSE)
wr <- read.csv('proj_wr15.csv', header=TRUE, stringsAsFactors=FALSE)

# generate unique list of column names
cols <- unique(c(names(k), names(qb), names(rb), names(te), names(wr)))
k[,'pos'] <- 'k'
qb[,'pos'] <- 'qb'
rb[,'pos'] <- 'rb'
te[,'pos'] <- 'te'
wr[,'pos'] <- 'wr'
cols <- c(cols, 'pos')

# create common columns in each data.frame
# initialize values to zero
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

# combine data.frames by row, using consistent column order
data <- rbind(k[,cols], qb[,cols], rb[,cols], te[,cols], wr[,cols])

### step 2
for(i in 3:18){
    data[data[,'Team'] %in% c('CLE', 'NO', 'NYG', 'PIT'),][i] <- data[data[,'Team'] %in% c('CLE', 'NO',
    data[!data[,'Team'] %in% c('CLE', 'NO', 'NYG', 'PIT'),][i] <- data[!data[,'Team'] %in% c('CLE', 'NO
}

### step 3
data <- data[order(-data['fpts']),] # sort and order the data by the fpts column descendingly
k20 <- data[data$pos=='k',][1:20,]
qb20 <- data[data$pos=='qb',][1:20,]
rb40 <- data[data$pos=='rb',][1:40,]
wr40 <- data[data$pos=='wr',][1:60,]
te20 <- data[data$pos=='te',][1:20,]
data <- rbind(k20, qb20, rb40, wr40, te20) # the projection data have 160 rows and 19 columns
```

```r
### step 4
NFL_15 <- read.csv('nfl_current15.csv', header=TRUE, stringsAsFactors=FALSE) # read in the observed dat

### step 5
NewData <- merge(data, NFL_15, by.x="PlayerName", by.y="Name", all.x=T) # merge projected and observed
NewData[is.na(NewData)] <- 0 # set missing values to zero

### step 6
Name=c('field goals','field goals attempted','extra points','passing attempts','passing completions',
                      'passing yards','passing touchdowns','passing interceptions','rushing attempts','
                      'rushing touchdowns','receiving attempts','receiving yards','receiving touchdowns
projected_col=c('fg','fga','xpt','pass_att','pass_cmp','pass_yds','pass_tds','pass_ints',
                           'rush_att','rush_yds','rush_tds','rec_att','rec_yds','rec_tds','fumbles')
observed_col=c("FGM","FGA","XPM","Att.pass","Cmp.pass","Yds.pass","TD.pass","Int.pass",
                              "Att.rush","Yds.rush","TD.rush","Rec.catch","Yds.catch","TD.catch","Fmb")

residue <- data.frame(matrix(ncol = 16, nrow = 160))
colnames(residue) <- c('pos', projected_col) # choose the colname of projected data for the residue
residue$pos = NewData$pos
for(i in 1:15){
    residue[projected_col[i]] = NewData[observed_col[i]] - NewData[projected_col[i]]
    # difference between observed and projected
}

noise <- split(residue, f=residue$pos)
for(i in 1:5){
    noise[[i]]$pos <- NULL # remove column of "pos"
}

### noise is the residue list with five elements. Each element is a data.frame with 15 columns of one '
# to check the residues, just enter noise
# noise
```

Task 2: Creating League S3 Class (80 points)

Create an S3 class called `league`. Place all code at the end of the instructions.

1. Create a function `league` that takes 5 arguments (`stats`, `nTeams`, `cap`, `posReq`, `points`). It should return an object of type `league`. Note that all arguments should remain attributes of the object. They define the league setup and will be needed to calculate points and dollar values.

2. Create a function `calcPoints` that takes 1 argument, a league object. It will modify the league object by calculating the number of points each player earns, based on the league setup.

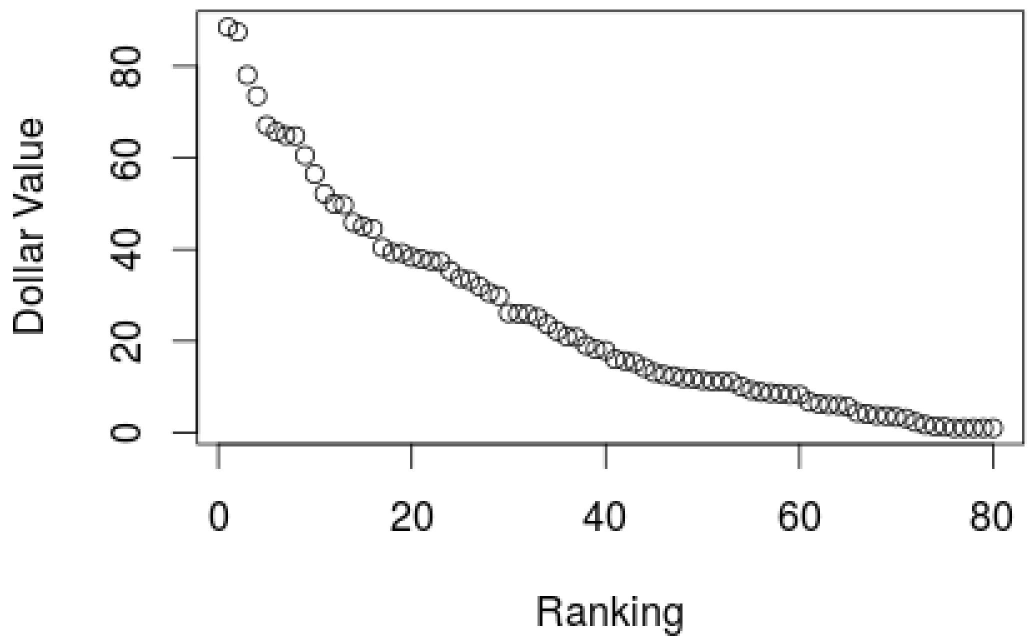
3. Create a function `buildValues` that takes 1 argument, a league object. It will modify the league object by calculating the dollar value of each player.

   As an example if a league has ten teams and requires one kicker, the tenth best kicker should be worth \$1. All kickers with points less than the 10th kicker should have dollar values of \$0.

4. Create a `print` method for the league class. It should print the players and dollar values (you may choose to only include players with values greater than \$0).

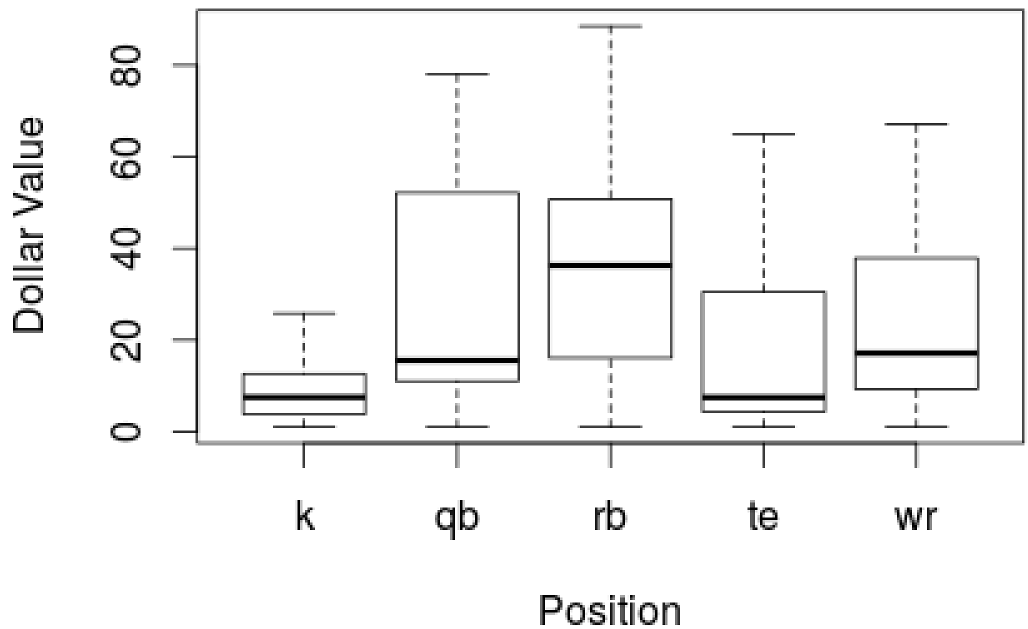
5. Create a `plot` method for the league class. Add minimal plotting decorations (such as axis labels).

- Here's an example:



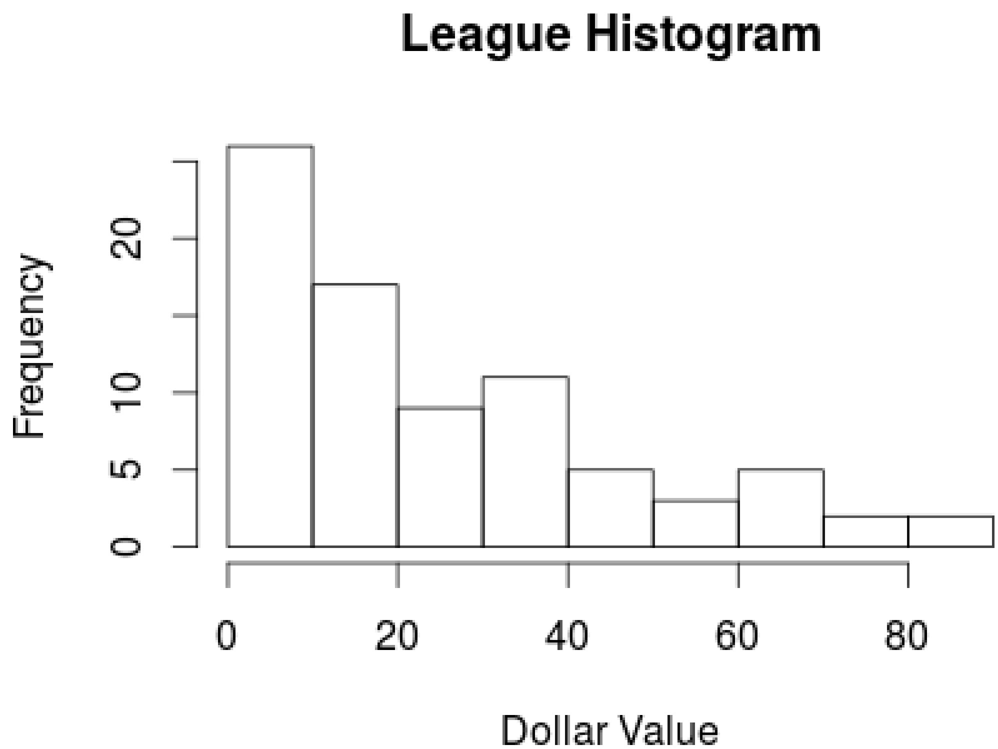
6. Create a `boxplot` method for the league class. Add minimal plotting decorations.

   - Here's an example:



7. Create a `hist` method for the league class. Add minimal plotting decorations.

    - Here's an example:



I will test your code with the following:

```
# x is combined projection data
pos <- list(qb=1, rb=2, wr=3, te=1, k=1)
pnts <- list(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)
l <- league(stats=x, nTeams=10, cap=200, posReq=pos, points=pnts)
l
hist(l)
boxplot(l)
plot(l)
```

I will test your code with additional league settings (using the same projection data). I will try some things that should work and some things that should break. Don't be too concerned, but here's some things I might try:

- Not including all positions
- Including new positions that don't exist
- Requiring no players at a posit

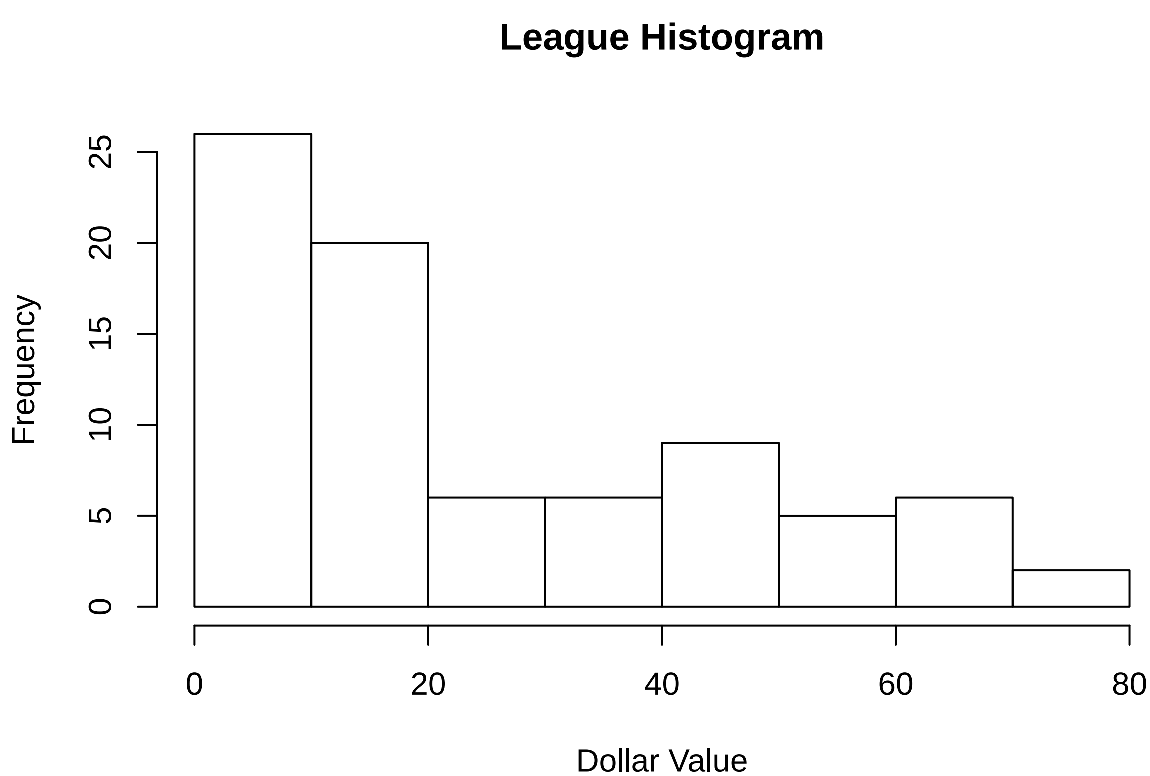
ion * Requiring too many players at a position (ie - there aren't 100 kickers)

Note that at this point it should be easy to change a league setting (such as `nTeams`) and re-run `calcPoints` and `buildValues`.

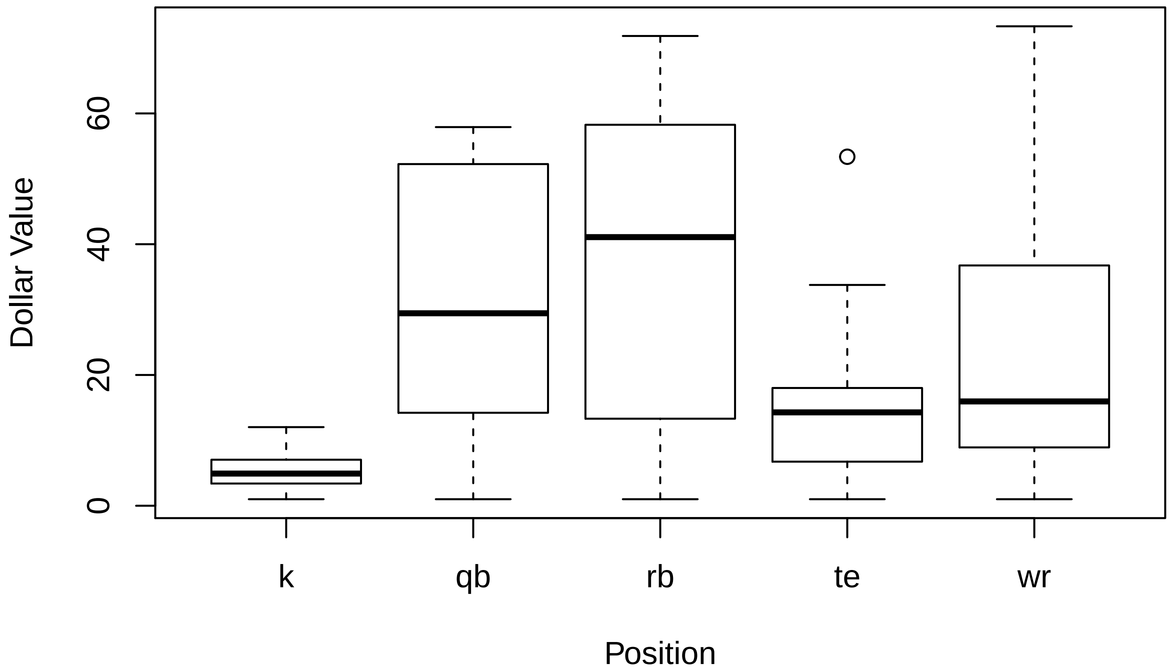
```
### step1
league <- function(stats, nTeams, cap, posReq, points){
    me <- list(stats=stats, nTeams=nTeams, cap=cap,
               posReq=posReq, points=points)
    me1 <- calcPoints(me)
    me2 <- buildValues(me1)
    class(me2) <- "league"
    return(me2)
}

### step2
calcPoints <- function(obj){
    obj$stats$earn <- obj$stats$fg * obj$points$fg + obj$stats$xpt * obj$points$xpt + obj$stats$pass_yds
    obj$stats$pass_tds * obj$points$pass_tds + obj$stats$pass_ints * obj$points$pass_ints + obj$stats$ru
    obj$stats$rush_tds * obj$points$rush_tds + obj$stats$fumbles * obj$points$fumbles + obj$stats$rec_yo
    obj$stats$rec_tds * obj$points$rec_tds
    return(obj)
}

### step3
buildValues <- function(obj){
    df <- obj$stats
    nTeams <- obj$nTeams
    cap <- obj$cap
    posReq <- obj$posReq
    x1 <- df[order(df[,'earn'], decreasing=TRUE),]
    for(i in names(posReq)){
        ix <- which(x1[,'pos'] == i)
        baseline <- posReq[[i]]*nTeams
        if(baseline == 0){
            x1[ix, 'marg'] <- -1
        }
        else{
            x1[ix, 'marg'] <- x1[ix,'earn'] - x1[ix[baseline],'earn']
        }
    }
    x2 <- x1[x1[,'marg'] >= 0,]
    x2[,'value'] <- x2[,'marg']*(nTeams*cap-nrow(x2))/sum(x2[,'marg']) + 1
    x3 <- x2[order(x2[,'value'], decreasing=TRUE),]
    list(stats=x3, nTeams=obj$nTeams, cap=obj$cap, posReq=obj$posReq, points=obj$points)
}

### step4
print.league <- function(obj){
    df <- obj$stats
    print(df[,c('PlayerName','value')])
}

### step5
plot.league <- function(obj){
    df <- obj$stats
    posReq <- obj$posReq
    nTeams <- obj$nTeams
```

```r
    sum <- sum(unlist(posReq))
    total <- sum*nTeams
    plot(x=1:total, y=df$value, xlab = "Ranking", ylab = "Dollar Value")
}

### step6
boxplot.league <- function(obj){
    df <- obj$stats
    boxplot(value ~ pos,data=df, xlab = "Position", ylab = "Dollar Value")
}

### step7
hist.league <- function(obj){
    df <- obj$stats
    hist(df$value, xlab = "Dollar Value", ylab = "Frequency", main = "League Histogram")
}

# my own test are performed at the end of each step.
```

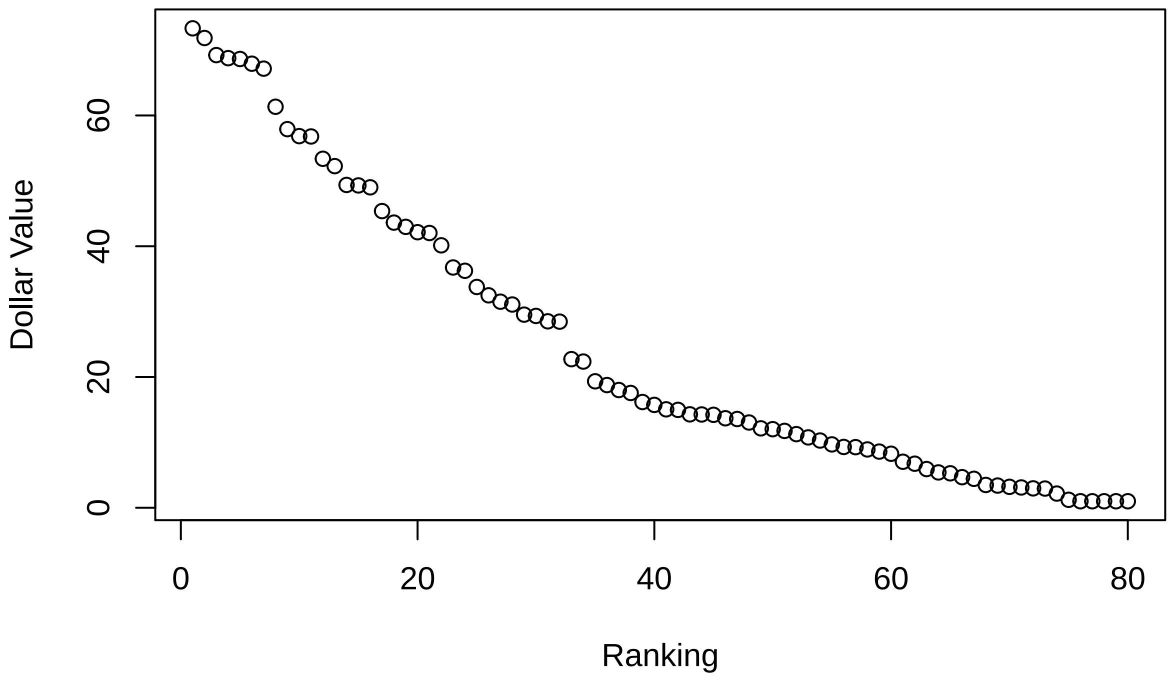
```r
# to comply with the test in the instruction
pos <- list(qb=1, rb=2, wr=3, te=1, k=1)
pnts <- list(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
             rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)

l <- league(stats=data, nTeams=10, cap=200, posReq=pos, points=pnts)
# l
hist(l)
```



```r
boxplot(l)
```




```r
plot(l)
```



Task 3: Simulations with Residuals (40 points)

Using residuals from task 1, create a list of league simulations. The simulations will be used to generate confidence intervals for player values. Place all code at the end of the instructions.

1. Create a function `addNoise` that takes 4 arguments: a league object, a list of residuals, number of simulations to generate, and a RNG seed. It will modify the league object by adding a new element `sims`, a matrix of simulated dollar values.

   The original league object contains a `stats` attribute. Each simulation will modify this by adding residual values. This modified `stats` data.frame will then be used to create a new league object (one for each simulation). Calculate dollar values for each simulation. Thus if 1000 simulations are requested, each player will have 1000 dollar values. Create a matrix of these simulated dollar values and attach it to the original league object.

   As an example assume you want to simulate new projections for quarterbacks. The residuals for quarterbacks is a 20x15 matrix. Each row from this matrix is no longer identified with a particular player, but rather it's potential error. Given the original projection for the first quarterback, sample one value between 1 and 20. Add the 15 columns from the sampled row to the 15 columns for the first quarterback. Repeat the process for every quarterback. Note that stats can't be negative so replace any negative values with 0.

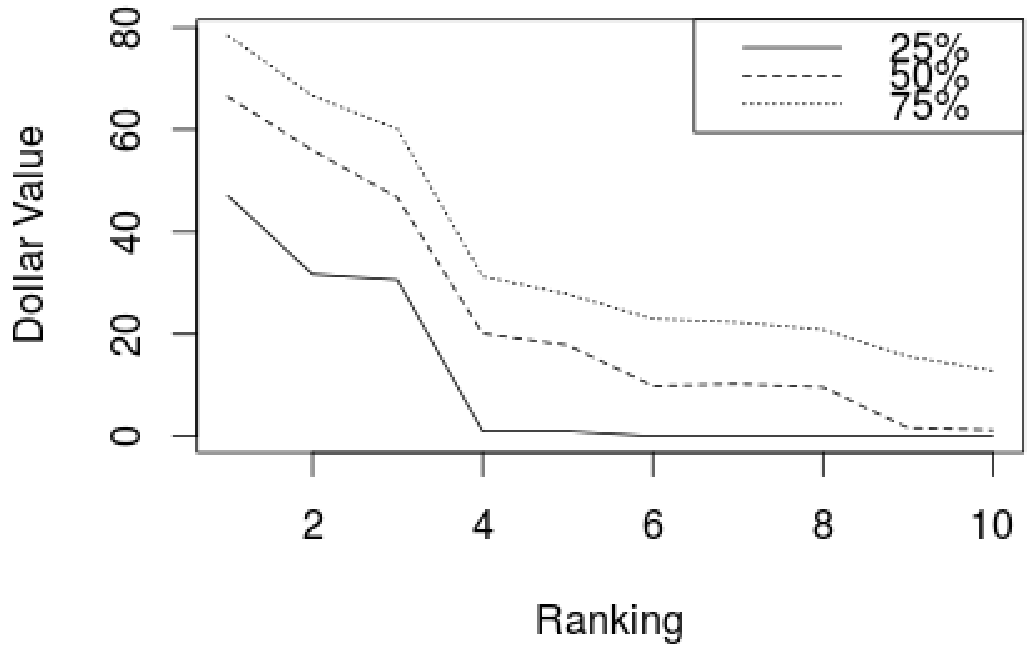
2. Create a `quantile` method for the league class; it takes at least two arguments, a league object and a probs vector. This method requires the `sims` element; it should fail if `sims` is not found. The `probs` vector should default to `c(0.25, 0.5, 0.75)`. It should run `quantile` on the dollar values for each player.

3. Create a function `conf.interval`; it takes at least two arguments, a league object and a probs vector. This method requires the `sims` element; it should fail if `sims` is not found. It should return a new object of type `league.conf.interval`.

   The new object will contain the output of `quantile`. However, results should be split by position and ordered by the last column (which should be the highest probability) descendingly. Restrict the number of rows to the number of required players at each position.

4. Create a `plot` method for the league.conf.interval class; it takes at least two arguments, a league.conf.interval object and a position. Plot lines for each probability; using the defaults, you would have three lines (0.25, 0.5, 0.75). Add minimal plotting decorations and a legend to distinguish each line.

   - Here's an example:



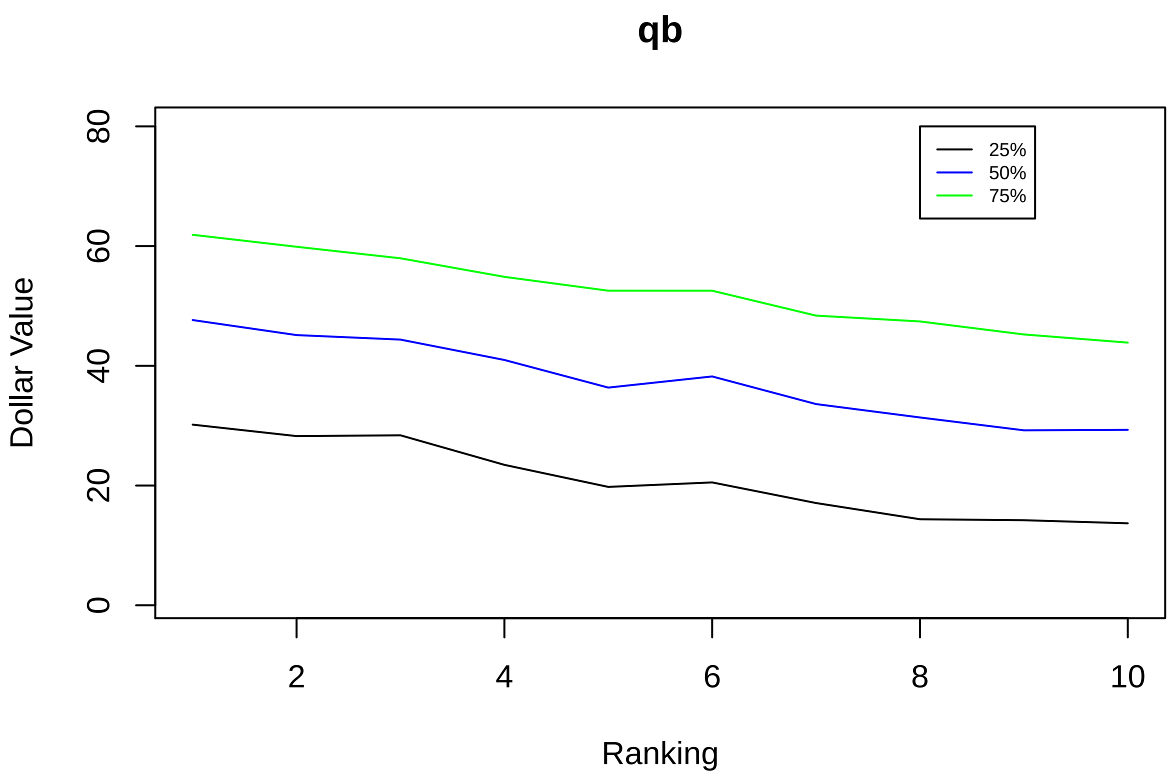
I will test your code with the following:

```
l1 <- addNoise(l, noise, 10000)
quantile(l1)
ci <- conf.interval(l1)
plot(ci, 'qb')
plot(ci, 'rb')
plot(ci, 'wr')
plot(ci, 'te')
plot(ci, 'k')
```

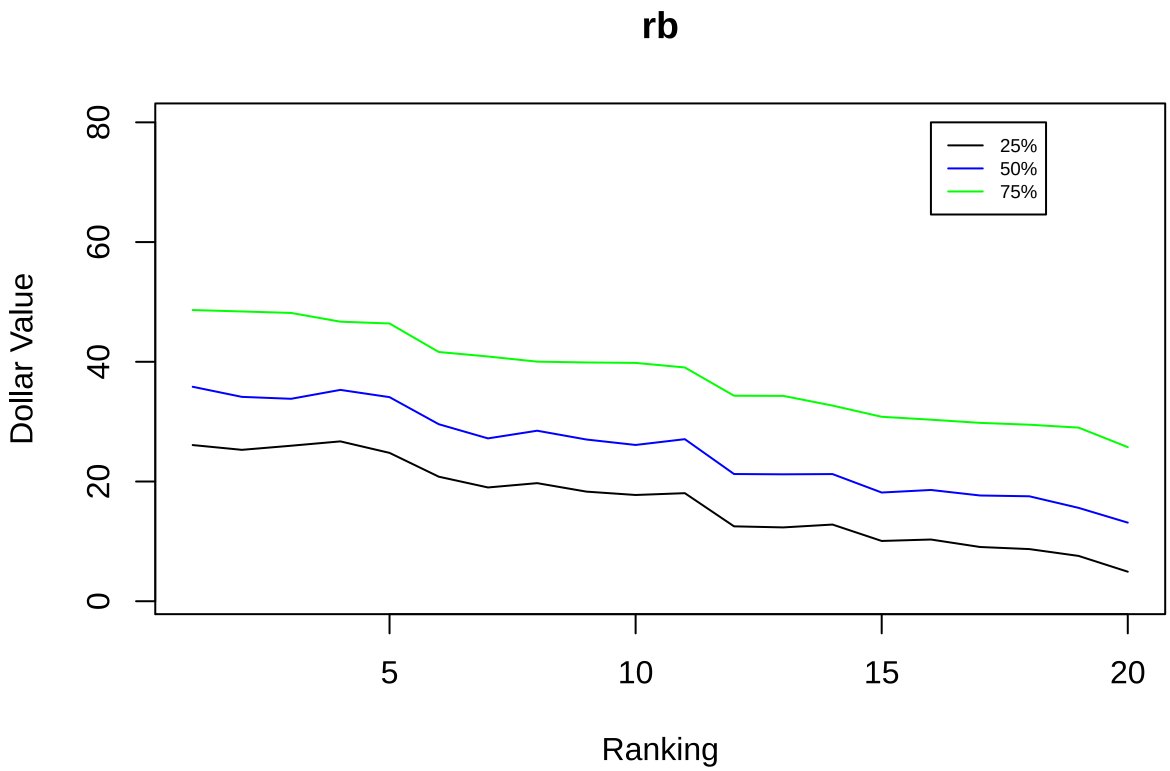
```
### step1
addNoise <- function(obj, residue, n=1000, seed=1){
    set.seed(seed)
    df <- obj$stats # get projected data you need for the simulation
    PlayerNum <- nrow(df)
    nTeams <- obj$nTeams
    cap <- obj$cap
    posReq <- obj$posReq
    points <- obj$points

    sims <- matrix(0, nrow = PlayerNum, ncol = n) # a matrix to store the simulation results
    pos <- c("k", "qb", "rb", "te", "wr") # all position names
    res_num <- list(k=20,qb=20,rb=40,te=20,wr=60) # number of residue candidates for each pos
    for(j in 1:n){ # simulate n times
```

```
        temp_df = df # get a copy of the projected data
        for(i in 1:PlayerNum){ # add residues to each Player
            temp_pos = df[i,"pos"] # the pos name of single player
            rand = sample(res_num[[temp_pos]],1) # the row number to be added
            for(name in names(residue[[temp_pos]])){
                temp_df[i,name] <- max(0, temp_df[i,name] + residue[[temp_pos]][rand,name])
            }
        }
        sim_stats = league(stats=temp_df, nTeams, cap, posReq, points)$stats  # new DataFrame with diff
        for(i in 1:PlayerNum){ # match PlayerName one by one
            sims[i,j] = sim_stats[sim_stats$PlayerName==df$PlayerName[i],"value"]
        }
    }
    me <- list(stats=df, nTeams=nTeams, cap=cap, posReq=posReq, points=points, sims=sims) # add element
    class(me) <- "league"
    return(me)
}

### step2
quantile.league <- function(obj, probs=c(0.25, 0.5, 0.75)){
    if(is.null(obj$sims)){
        stop("sims is not found!")
    }
    df = obj$stats
    sim = obj$sims
    PlayerNum <- nrow(df)
    output <- matrix(0, nrow = PlayerNum, ncol = 3)
    for(i in 1:PlayerNum){
        output[i, 1] = quantile(x=sim[i,], probs=probs[1])
        output[i, 2] = quantile(x=sim[i,], probs=probs[2])
        output[i, 3] = quantile(x=sim[i,], probs=probs[3])
    }
    return(output)
}

### step3
conf.interval <- function(obj, probs=c(0.25, 0.5, 0.75)){
    if(is.null(obj$sims)){
        stop("sims is not found!")
    }
    sim = quantile(obj, probs) # get simulation data
    df = obj$stats
    PlayerNum <- nrow(df)
    output <- data.frame(pos=NA,sim)
    output$pos = df$pos
    output = output[order(-output[,4]),]
    me = split(output, f=output$pos)
    for(i in 1:5){
        me[[i]]$pos <- NULL
    }
    class(me) <- "league.conf.interval"
    return(me)
}
```

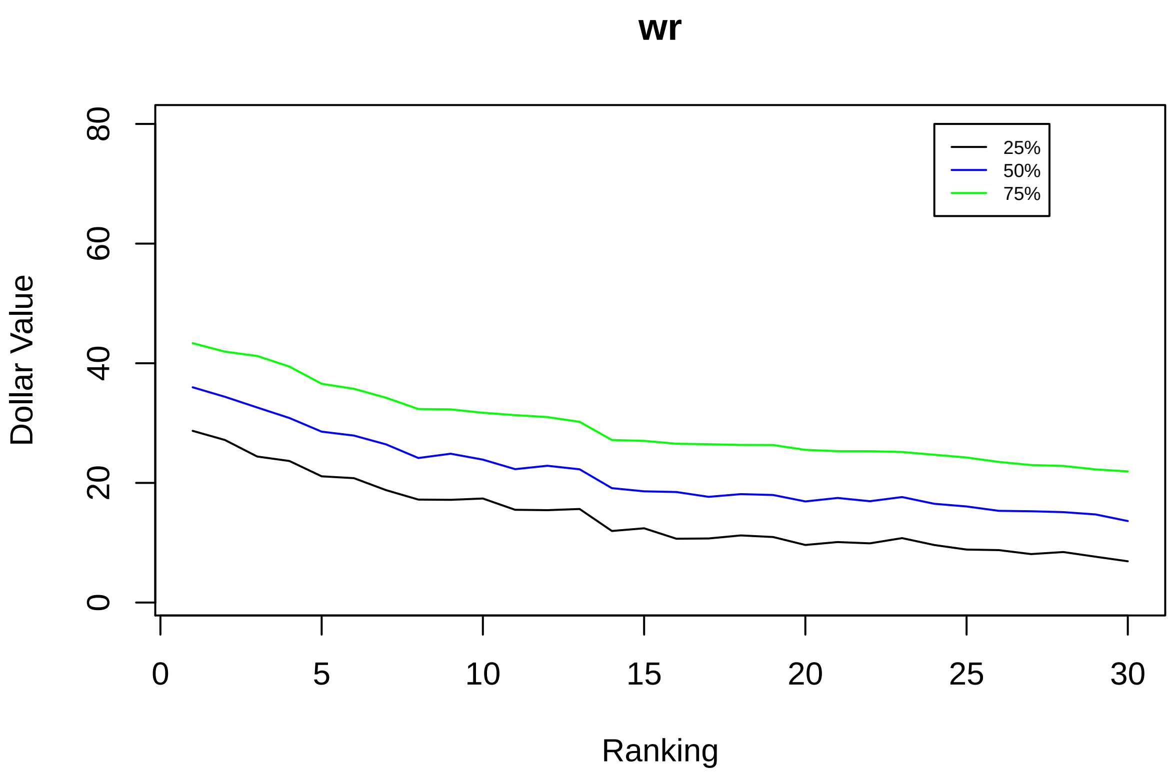
```r
### step4
plot.league.conf.interval <- function(obj, pos){
    matrix <- obj[[pos]]
    Row_num <- nrow(matrix)
    plot(1:Row_num,matrix[,1],type="l",xlim=c(1,Row_num),ylim=c(1,80),xlab="Ranking",ylab="Dollar Value
    lines(1:Row_num,matrix[,2],type="l",col = "blue")
    lines(1:Row_num,matrix[,3],type="l",col = "green")
    legend(0.8*Row_num,80,c("25%","50%","75%"),lty=c(1,1,1),lwd=c(1,1,1),col=c("black","blue","green"),
}

### test
l1 <- addNoise(l, noise, 1000) # changing 1000 to 500 or 100 if you do not want to wait for long.
# quantile(l1)
ci <- conf.interval(l1)
plot(ci, 'qb')
```



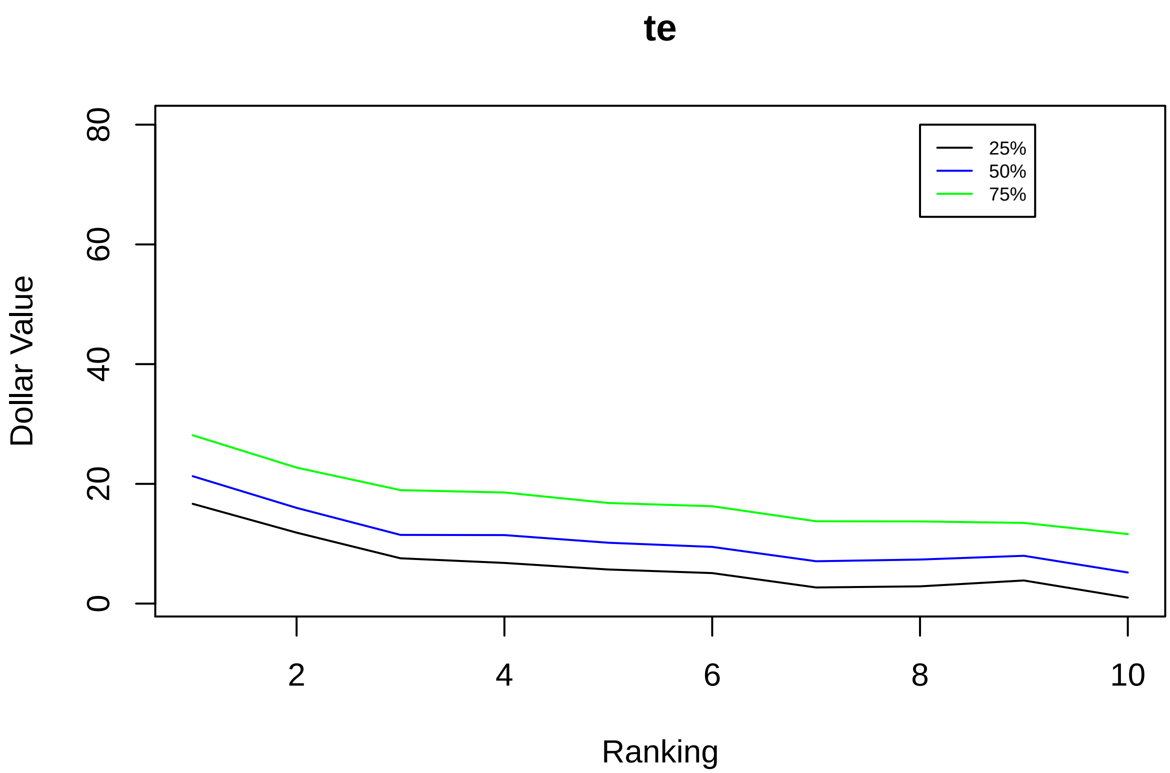

```r
plot(ci, 'rb')
```



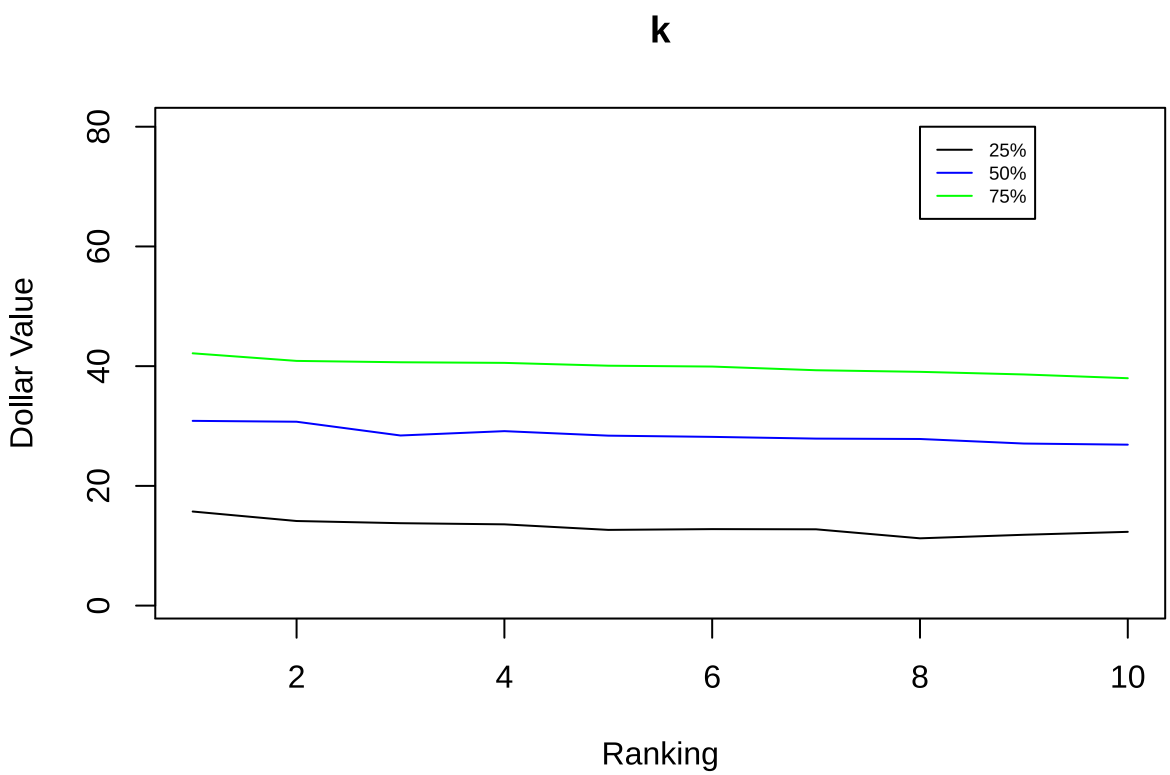

```
plot(ci, 'wr')
```




```
plot(ci, 'te')
```


te
Dollar Value
0
20
40
60
80
2
4
6
8
10
Ranking
25%
50%
75%

```r
plot(ci, 'k')
```


k
Dollar Value
0
20
40
60
80
2
4
6
8
10
Ranking
25%
50%
75%

Additional Tips

Use your best judgement in interpreting my instructions, and please do not hesitate to ask for clarification.

You have most of the code for tasks 1 and 2, it's a matter of restructuring it.

If you're stuck, explain your algorithm, why it fails, and move on. Attempt everything.