Bios 6301: Final Project

Wooyeol Lee

12/14/2015

Due Monday, 14 December, 6:00 PM

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.

```
library(plyr)

  path<- paste("C:/Users/Wooyeol/Dropbox/me/coursework/fall2015/statistical computing/final/")
setwd(path)
  ######## 1. read in CSV files
  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)
  ######## add position column
  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')
  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

  ###merging
  x <- rbind(k[,cols], qb[,cols], rb[,cols], te[,cols], wr[,cols])

  ######## 2.add percent game column
  x[,'perc'] <- 9/16      #### teams played 9 games

  cle <- which(x[,'Team']=='CLE')
  no <- which(x[,'Team']=='NO')
  nyg <- which(x[,'Team']=='NYG')
  pit <- which(x[,'Team']=='PIT')

  ten.game <- c(cle, no, nyg, pit)  ####row numbers of 10-game teams
  x[ten.game, 'perc'] <- 10/16    #### these team played 10 games

  ######## multiply by perc
  x[,3:18] <- x[,3:18]*x[,'perc']

    ######### 3. sort by ftp
  x2 <- x[order(x[,'fpts'], decreasing=TRUE),]
  ### subset data
  k <- x2[ which(x2$pos=='k'),]
  k <- k[1:20,]
  qb <- x2[ which(x2$pos=='qb'),]
  qb <- qb[1:20,]
  rb <- x2[ which(x2$pos=='rb'),]
  rb <- rb[1:40,]
  wr <- x2[ which(x2$pos=='wr'),]
  wr <- wr[1:60,]
  te <- x2[ which(x2$pos=='te'),]
  te <- te[1:20,]

  x2<- rbind(k,qb,rb,wr,te)
  x2 <- x2[c(-20)]                                ## drop 'perc'
  names(x2)[1] <- "Name"                          ## change name of variable

  ##########                                      ##NOTE: x2 is the projected data. Use this for Task2!

  ####### 4. read observed data
  observed <- read.csv("nfl_current15.csv")

  ######## 5. merge the projected data with the observed data by the player's name.
  total <- merge(x2,observed,by="Name", all.x=T)  ## merge
  total <- total[c(-20,-21)]                      ## drop redundant variables "team, pos"
  total[is.na(total)] <- 0                        ## replace missing data with 0

  ####### 6. take difference between observed and projected
  total[,'d_fg']<-total[,'FGM']-total[,'fg']
  total[,'d_fga']<-total[,'FGA']-total[,'fga']
  total[,'d_xpt']<-total[,'XPM']-total[,'xpt']
  total[,'d_pass_att']<-total[,'Att.pass']-total[,'pass_att']
  total[,'d_pass_cmp']<-total[,'Cmp.pass']-total[,'pass_cmp']
  total[,'d_pass_yds']<-total[,'Yds.pass']-total[,'pass_yds']
  total[,'d_pass_tds']<-total[,'TD.pass']-total[,'pass_tds']
  total[,'d_pass_ints']<-total[,'Int.pass']-total[,'pass_ints']
  total[,'d_rush_att']<-total[,'Att.rush']-total[,'rush_att']
  total[,'d_rush_yds']<-total[,'Yds.rush']-total[,'rush_yds']
  total[,'d_rush_tds']<-total[,'TD.rush']-total[,'rush_tds']
  total[,'d_rec_att']<-total[,'Rec.catch']-total[,'rec_att']
  total[,'d_rec_yds']<-total[,'Yds.catch']-total[,'rec_yds']
  total[,'d_rec_tds']<-total[,'TD.catch']-total[,'rec_tds']
  total[,'d_fumbles']<-total[,'Fmb']-total[,'fumbles']
  #### subset res. 15columns
  res<- total[,35:49]

  ### split data by position
  res.k <- res[which(total$pos=='k'),]
  res.qb <- res[which(total$pos=='qb'),]
  res.rb <- res[which(total$pos=='rb'),]
  res.wr <- res[which(total$pos=='wr'),]
  res.te <- res[which(total$pos=='te'),]

  ### This is the data.
  dat <-list(res.k, res.qb, res.rb, res.wr, res.te)                     ####NOTE: Use it for Task3!
  names(dat)<- c("res.k", "res.qb", "res.rb", "res.wr", "res.te")
```

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.

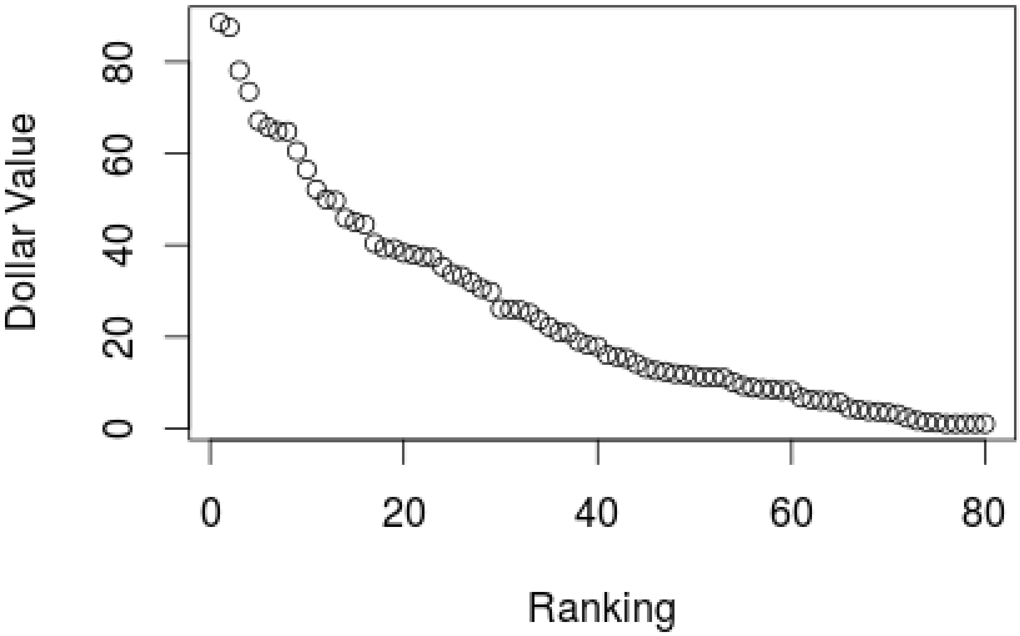
3. Create a function `buildValues` that takes 1 argument, a league object. It will modify the league object by calculating the dollar value of each player.

   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).

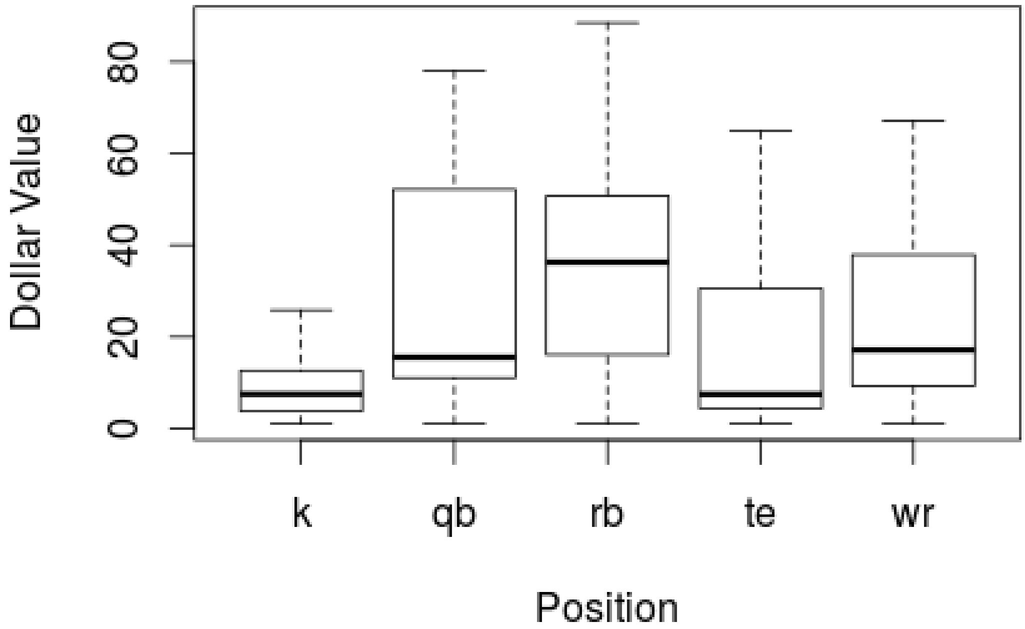
5. Create a `plot` method for the league class. Add minimal plotting decorations (such as axis labels).

   - Here's an example:



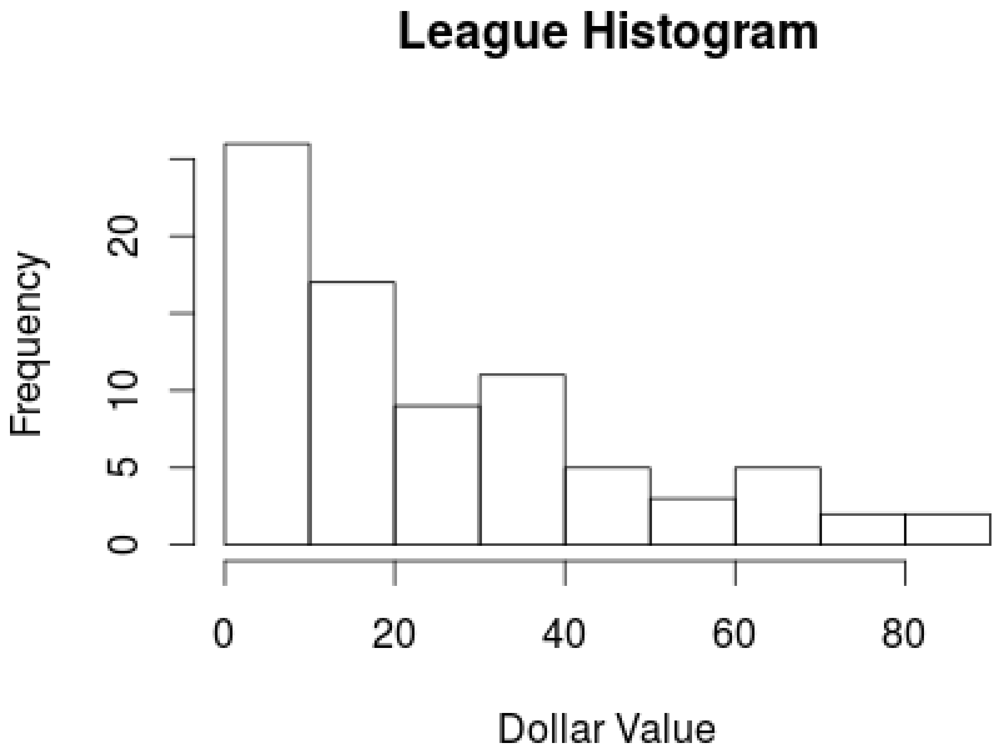
6. Create a `boxplot` method for the league class. Add minimal plotting decorations.

   - 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:

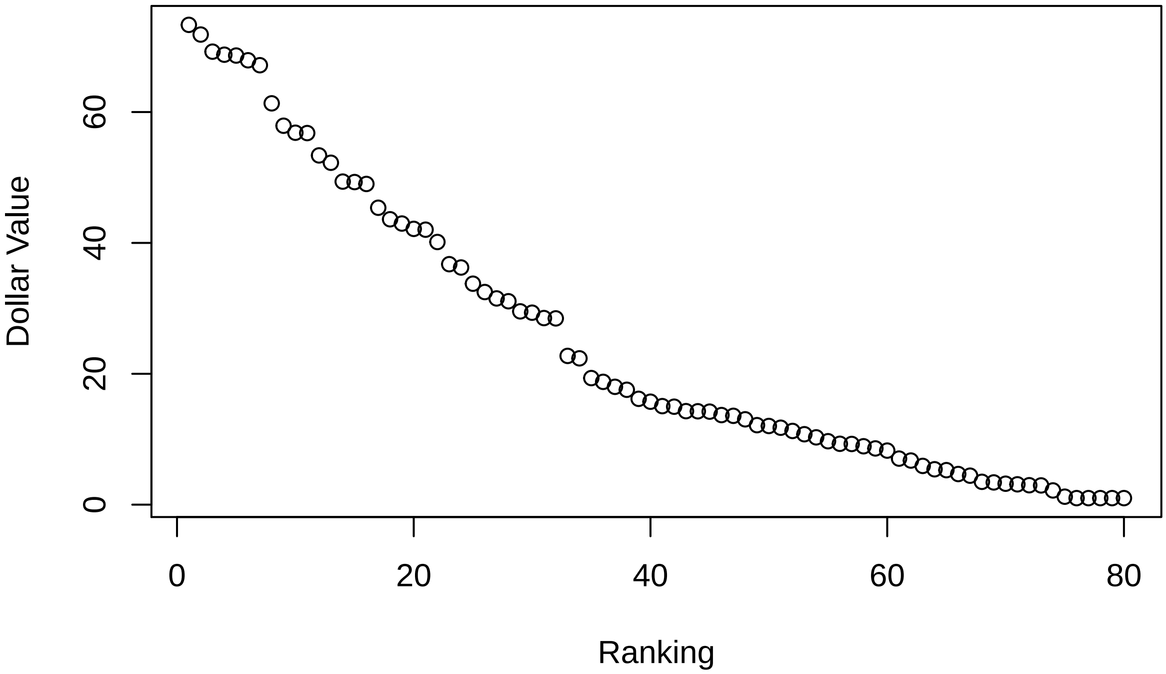
- Not including all positions
- Including new positions that don't exist
- Requiring no players at a position
- Requiring too many players at a position (ie - there aren't 100 kickers)

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

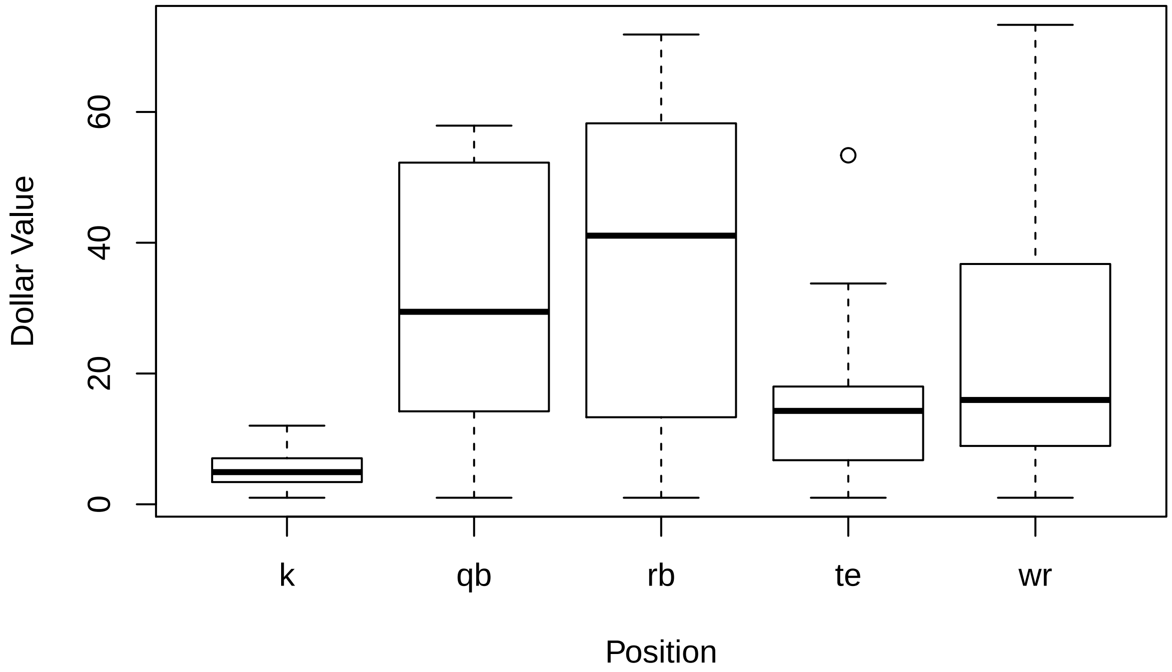
```
  ###Task 2: creating League S3class

  ###1. league setup.
 league<- function(stats, nTeams, cap, posReq, points) {
   x<-list(stats, nTeams, cap, posReq, points)
   class(x) <-c("league")
   names(x) <- c('stats','nTeams','cap','posReq','points')
   return(x)
   }
  ###2. Create a function calcPoints that takes 1 argument, a league object.
 calcPoints<- function(league) {
   league$stats[,'p_fg'] <- league$stats[,'fg']*league$points$fg
   league$stats[,'p_xpt'] <- league$stats[,'xpt']*league$points$xpt
   league$stats[,'p_pass_yds'] <- league$stats[,'pass_yds']*league$points$pass_yds
   league$stats[,'p_pass_tds'] <- league$stats[,'pass_tds']*league$points$pass_tds
   league$stats[,'p_pass_ints'] <- league$stats[,'pass_ints']*league$points$pass_ints
   league$stats[,'p_rush_yds'] <- league$stats[,'rush_yds']*league$points$rush_yds
   league$stats[,'p_rush_tds'] <- league$stats[,'rush_tds']*league$points$rush_tds
   league$stats[,'p_fumbles'] <- league$stats[,'fumbles']*league$points$fumbles
   league$stats[,'p_rec_yds'] <- league$stats[,'rec_yds']*league$points$rec_yds
   league$stats[,'p_rec_tds'] <- league$stats[,'rec_tds']*league$points$rec_tds
   return(league)
 }


  ###3. Create a function buildValues that takes 1 argument, a league object.
buildValues<- function(league) {

   # this is total fantasy points for each player
   league$stats[,'points'] <- rowSums(league$stats[,grep("^p_", names(league$stats))])

   # create new data.frame ordered by points descendingly
   league2 <- league$stats[order(league$stats[,'points'], decreasing=TRUE),]

   # determine the row indeces for each position
   k.ix <- which(league2[,'pos']=='k')
   qb.ix <- which(league2[,'pos']=='qb')
   rb.ix <- which(league2[,'pos']=='rb')
   te.ix <- which(league2[,'pos']=='te')
   wr.ix <- which(league2[,'pos']=='wr')

   # calculate marginal points by subtracting "baseline" player's points
   league2[k.ix, 'marg'] <- league2[k.ix,'points'] - league2[k.ix[league$nTeams*league$posReq$k],'points
   league2[qb.ix, 'marg'] <- league2[qb.ix,'points'] - league2[qb.ix[league$nTeams*league$posReq$qb],'po
   league2[rb.ix, 'marg'] <- league2[rb.ix,'points'] - league2[rb.ix[league$nTeams*league$posReq$rb],'po
   league2[te.ix, 'marg'] <- league2[te.ix,'points'] - league2[te.ix[league$nTeams*league$posReq$te],'po
   league2[wr.ix, 'marg'] <- league2[wr.ix,'points'] - league2[wr.ix[league$nTeams*league$posReq$wr],'po

   # create a new data.frame subset by non-negative marginal points
   league3 <- league2[league2[,'marg'] >= 0,]

   # re-order by marginal points
   league3 <- league3[order(league3[,'marg'], decreasing=TRUE),]
   # reset the row names
   rownames(league3) <- NULL

   # calculation for player value
   league3[,'value'] <- league3[,'marg']*(league$nTeams*league$cap-nrow(league3))/sum(league3[,'marg'])

   # create a data.frame with more interesting columns
   league$stats <- league3[,c('Name','pos','points','marg','value')]

##
   return(league)
}


   ###4. Create a print method for the league class.
 print.league <- function(league) {
   table <- league$stats[,c(1,5)]
   return(table)
   }

  ###5. Create a plot method for the league class.
 plot.league <- function(league) {
   y<- league$stats$value
   x<- seq(league$stats$value)
   plot(y~x, ylab="Dollar Value", xlab="Ranking")
 }


 ###6. Create a boxplot method for the league class.
 boxplot.league <- function(league) {
   y<- league$stats$value
   x<- as.factor(league$stats$pos)
   boxplot(y~x, ylab="Dollar Value", xlab="Position")
 }

 ###7. Create a hist method for the league class.
 hist.league <- function(league) {
   y<- league$stats$value
   hist(y, ylab="Frequency", xlab="Dollar Value", main="League Histogram")
 }


 #He will test with this: 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=x2, nTeams=10, cap=200, posReq=pos, points=pnts)

 a<- calcPoints(l)
 a2<- buildValues(a)
 print(a2)
```

```
##                  Name     value
## 1       Antonio Brown 73.320335
## 2      Marshawn Lynch 71.841085
## 3        Le'Veon Bell 69.227636
## 4   Odell Beckham Jr. 68.762241
## 5     Adrian Peterson 68.630184
## 6          Eddie Lacy 67.911326
## 7      Jamaal Charles 67.154129
## 8    Demaryius Thomas 61.326585
## 9          Drew Brees 57.907162
## 10        Andrew Luck 56.838992
## 11         Dez Bryant 56.783400
## 12     Rob Gronkowski 53.371220
## 13      Aaron Rodgers 52.249801
## 14      C.J. Anderson 49.369576
## 15     Calvin Johnson 49.297690
## 16       Randall Cobb 49.005354
## 17         Matt Forte 45.372724
## 18       LeSean McCoy 43.613917
## 19        Julio Jones 42.957360
## 20     DeMarco Murray 42.142654
## 21        Jeremy Hill 42.022844
## 22        Mark Ingram 40.137838
## 23     Alshon Jeffery 36.746425
## 24         A.J. Green 36.243224
## 25       Jimmy Graham 33.765559
## 26         Mike Evans 32.490784
## 27     Russell Wilson 31.514095
## 28      Brandin Cooks 31.083419
## 29     Peyton Manning 29.537714
## 30 Ben Roethlisberger 29.338138
## 31   Emmanuel Sanders 28.513102
## 32        T.Y. Hilton 28.465178
## 33       Lamar Miller 22.733481
## 34     Justin Forsett 22.359675
## 35      Alfred Morris 19.340470
## 36    Jordan Matthews 18.765384
## 37         Greg Olsen 18.008186
## 38       Travis Kelce 17.557702
## 39    Martavis Bryant 16.173767
## 40    DeAndre Hopkins 15.731802
## 41     Julian Edelman 15.056075
## 42          Matt Ryan 14.970771
## 43  Martellus Bennett 14.289293
## 44       Jason Witten 14.270123
## 45        Eli Manning 14.215490
## 46      Andre Johnson 13.690244
## 47      Melvin Gordon 13.575227
## 48        Carlos Hyde 13.043272
## 49     DeSean Jackson 12.147095
## 50 Stephen Gostkowski 12.022493
## 51         Frank Gore 11.758912
## 52      Davante Adams 11.265296
## 53      Sammy Watkins 10.762095
## 54    Garrett Hartley 10.286584
## 55        Golden Tate  9.693392
## 56    Latavius Murray  9.300417
## 57      Julius Thomas  9.271662
## 58      Jeremy Maclin  8.921818
## 59       Keenan Allen  8.591143
## 60   Brandon Marshall  8.260468
## 61      Justin Tucker  7.038409
## 62       Dwayne Allen  6.741281
## 63         Josh Brown  5.920185
## 64    Steven Hauschka  5.408997
## 65         Cam Newton  5.272893
## 66          Zach Ertz  4.680554
## 67     Dustin Hopkins  4.429220
## 68        Cody Parkey  3.492042
## 69       Connor Barth  3.396194
## 70    Marques Colston  3.202369
## 71       Mason Crosby  3.108651
## 72       Mike Wallace  2.969672
## 73    Vincent Jackson  2.940917
## 74       Amari Cooper  2.174135
## 75      Joseph Randle  1.220450
## 76          Tony Romo  1.000000
## 77     Adam Vinatieri  1.000000
## 78    Rashad Jennings  1.000000
## 79        Eric Decker  1.000000
## 80       Coby Fleener  1.000000
```

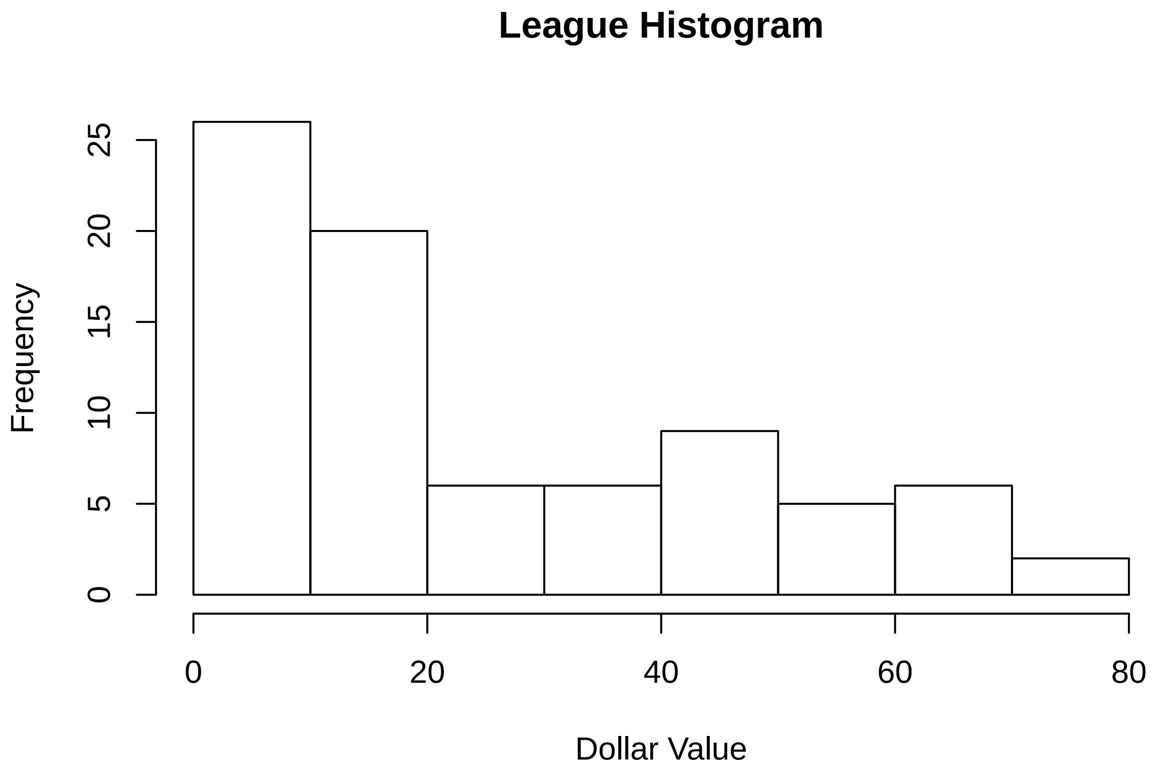
```
plot(a2)
```



```
boxplot(a2)
```



```
hist(a2)
```



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.

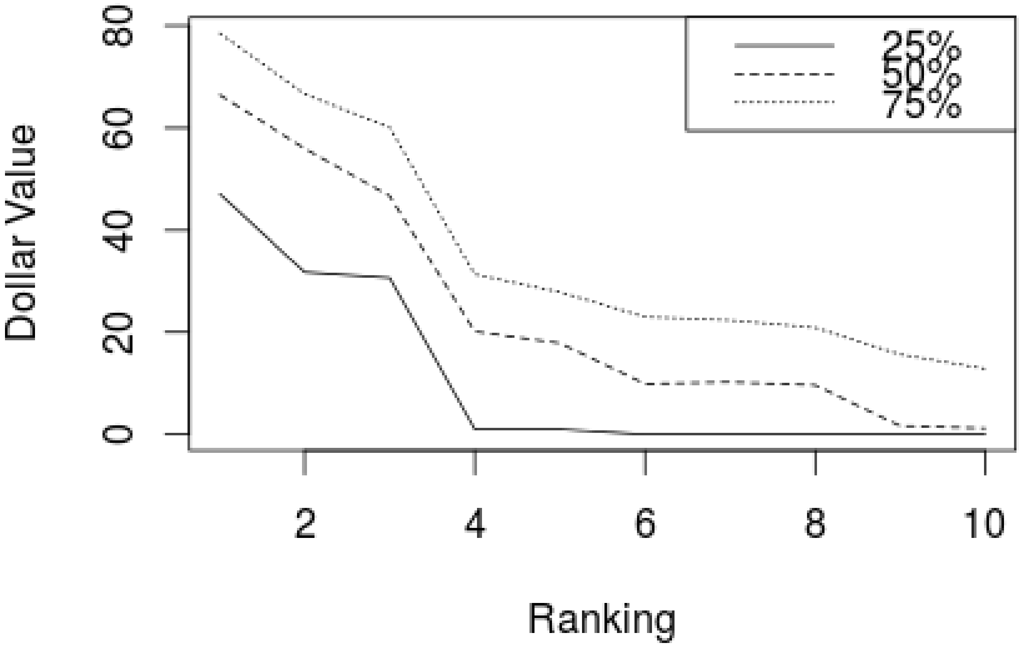
2. Create a `quantile` method for the league class; it takes at least two arguments, a league object and a probs vector. This method requires the `sims` element; it should fail if `sims` is not found. The `probs` vector should default to `c(0.25, 0.5, 0.75)`. It should run `quantile` on the dollar values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

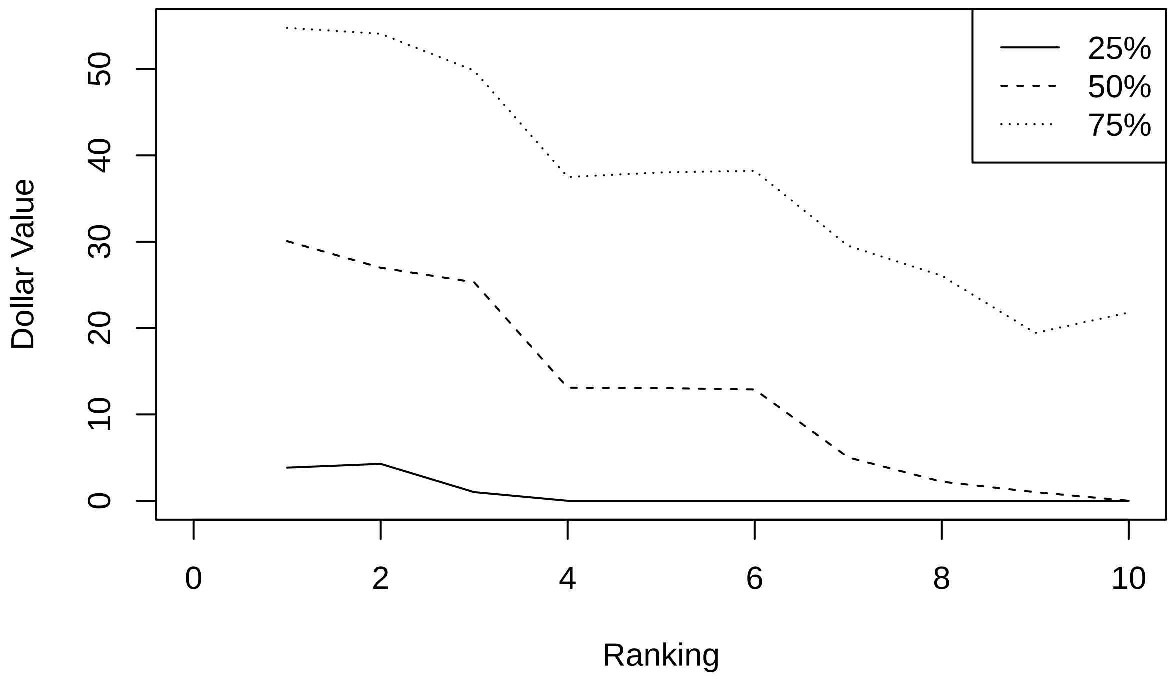
```
l1 <- addNoise(l, noise, 10000)
quantile(l1)
ci <- conf.interval(l1)
plot(ci, 'qb')
plot(ci, 'rb')
plot(ci, 'wr')
plot(ci, 'te')
plot(ci, 'k')
```

```
########## Task 3 Simulations with Residuals

 ## 1. Create a function addNoise

addNoise <- function(league, resid, nsim, seed=sample(1:10000,1)) {
  set.seed(seed)
  league$sims <- as.data.frame(matrix(0, ncol = nrow(league$stats), nrow = 0))
  colnames(league$sims) <- l$stats$Name    ## column names are players' name
  s <- league$stats  #copy of initial data

  for(i in 1:nsim) {

    #k
    k<-s[which(s[,'pos']=="k"),]                                  ##subset k
    noise<-sample(1:nrow(resid$res.k), nrow(k), replace=T)   ##sample sequence of noise (w/replace)

    k[,"fg"]<- k[,"fg"]+resid$res.k[noise,"d_fg"]
    k[,"xpt"]<- k[,"xpt"]+resid$res.k[noise,"d_xpt"]

    #qb
    qb<-s[which(s[,'pos']=="qb"),]                                   ##subset qb
    noise<-sample(1:nrow(resid$res.qb), nrow(qb), replace=T)    ##sample sequence of noise (w/replace)

    qb[,"pass_yds"]<- qb[,"pass_yds"]+resid$res.qb[noise,"d_pass_yds"]
    qb[,"pass_tds"]<- qb[,"pass_tds"]+resid$res.qb[noise,"d_pass_tds"]
    qb[,"pass_ints"]<- qb[,"pass_ints"]+resid$res.qb[noise,"d_pass_ints"]
    qb[,"rush_yds"]<- qb[,"rush_yds"]+resid$res.qb[noise,"d_rush_yds"]
    qb[,"rush_tds"]<- qb[,"rush_tds"]+resid$res.qb[noise,"d_rush_tds"]
    qb[,"fumbles"]<- qb[,"fumbles"]+resid$res.qb[noise,"d_fumbles"]

    #rb
    rb<-s[which(s[,'pos']=="rb"),]                                   ##subset rb
    noise<-sample(1:nrow(resid$res.rb), nrow(rb), replace=T)    ##sample sequence of noise (w/replace)

    rb[,"rush_yds"]<- rb[,"rush_yds"]+resid$res.rb[noise,"d_rush_yds"]
    rb[,"rush_tds"]<- rb[,"rush_tds"]+resid$res.rb[noise,"d_rush_tds"]
    rb[,"fumbles"]<- rb[,"fumbles"]+resid$res.rb[noise,"d_fumbles"]
    rb[,"rec_yds"]<- rb[,"rec_yds"]+resid$res.rb[noise,"d_rec_yds"]
    rb[,"rec_tds"]<- rb[,"rec_tds"]+resid$res.rb[noise,"d_rec_tds"]

    #te
    te<-s[which(s[,'pos']=="te"),]                                   ##subset te
    noise<-sample(1:nrow(resid$res.te), nrow(te), replace=T)    ##sample sequence of noise (w/replace)

    te[,"fumbles"]<- te[,"fumbles"]+resid$res.te[noise,"d_fumbles"]
    te[,"rec_yds"]<- te[,"rec_yds"]+resid$res.te[noise,"d_rec_yds"]
    te[,"rec_tds"]<- te[,"rec_tds"]+resid$res.te[noise,"d_rec_tds"]

    #wr
    wr<-s[which(s[,'pos']=="wr"),]                                    ##subset wr
    noise<-sample(1:nrow(resid$res.wr), nrow(wr), replace=T)     ##sample sequence of noise (w/replace)

    wr[,"rush_yds"]<- wr[,"rush_yds"]+resid$res.wr[noise,"d_rush_yds"]
    wr[,"rush_tds"]<- wr[,"rush_tds"]+resid$res.wr[noise,"d_rush_tds"]
    wr[,"fumbles"]<- wr[,"fumbles"]+resid$res.wr[noise,"d_fumbles"]
    wr[,"rec_yds"]<- wr[,"rec_yds"]+resid$res.wr[noise,"d_rec_yds"]
    wr[,"rec_tds"]<- wr[,"rec_tds"]+resid$res.wr[noise,"d_rec_tds"]

    rev.stat<- rbind(k,qb,rb, te, wr)
    rev.stat[rev.stat<0]=0                                       ##stat cannot be negative ->0
    ###
    league$stats <- rev.stat                                     ## stat with noise
    league<-calcPoints(league)                                   ## calculate points

    league<-buildValues(league)                                  ## calculate dollar values
    rep<- as.data.frame(t(league$stats$value))        ## 1*nperson data.frame
     colnames(rep) <- league$stats$Name               ## column names are players' name

    league$sims<-rbind.fill(league$sims, rep)   ##package plyr because of missing data
    league$sims[is.na(league$sims)] <- 0        ##missing values are 0 dollar.
  }
  league$stats <- s                             #paste initial data
   return(league)
}


 ## 2. Create a quantile method for the league class
quantile.league <- function(league, prob=c(0.25, 0.5, 0.75)) {
  if(is.null(league$sims) == T) {
    stop("No sim data found")
    }
  apply(league$sims,2,function(x) quantile(x,prob=prob))
}

 ## 3. Create a function conf.interval
conf.interval <- function(league, prob=c(0.25, 0.5, 0.75)) {
  if(is.null(league$sims) == T) {
    stop("No sim data found")
    }
    dat<- as.data.frame(quantile(league))
    l<- as.numeric(dat[1,])
    m<- as.numeric(dat[2,])
    u<- as.numeric(dat[3,])
    dat<-data.frame(l,m,u, league$stats$Name, league$stats$pos)
    dat<- dat[order(dat[,2], decreasing=T),]   ### sorting by 50% quantile

    colnames(dat)<-c("25%","50%","75%","Name","Position")

    #k
    k<- dat[which(dat$Position=='k'),][seq(league$nTeams*league$posReq$k),]
    #qb
    qb<- dat[which(dat$Position=='qb'),][seq(league$nTeams*league$posReq$qb),]
    #rb
    rb<- dat[which(dat$Position=='rb'),][seq(league$nTeams*league$posReq$rb),]
    #te
    te<- dat[which(dat$Position=='te'),][seq(league$nTeams*league$posReq$te),]
```

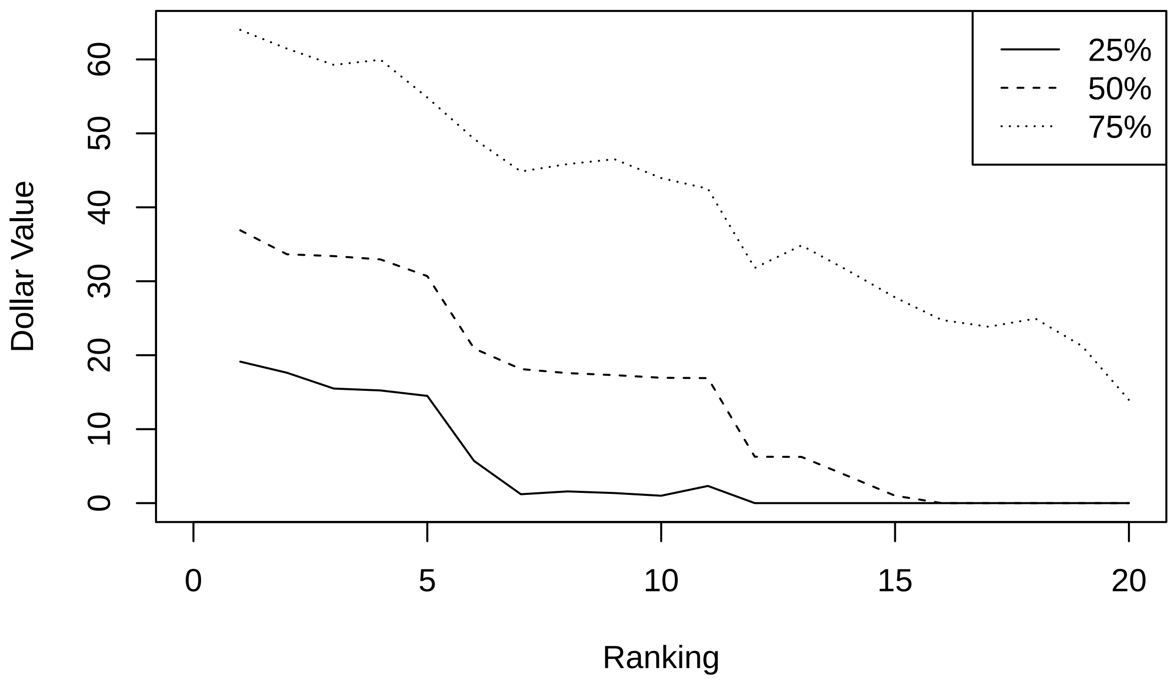
```r
    #wr
    wr<- dat[which(dat$Position=='wr'),][seq(league$nTeams*league$posReq$wr),]
    ## make a list
    x<-list(k, qb, rb, te, wr)
    names(x) <- c('k','qb','rb','te','wr')
    class(x)<-'conf.interval'
  return(x)
}

 ## 4. Create a plot method for the league.conf.interval class
plot.conf.interval <- function(ci, pos) {
  pos<-substitute(pos)
  dat<-as.data.frame(ci[pos])
 x<-seq_along(dat[,2])

plot(NULL,ylab="Dollar Value", xlab="Ranking", ylim=c(0,max(dat[,3])), xlim=c(0,length(x)))
lines(dat[,1]~x, lty="solid")
lines(dat[,2]~x, lty="dashed")
lines(dat[,3]~x, lty="dotted")
legend("topright", lty=c("solid","dashed","dotted"), c("25%","50%","75%"))
}


## he will test with this.

noise<- dat
l1 <- addNoise(l, noise, 1000)
quantile(l1)
```

```
##     Stephen Gostkowski Garrett Hartley Dustin Hopkins Justin Tucker
## 25%           0.000000        0.000000        0.00000       0.00000
## 50%           5.616695        4.334068        1.00000       1.00000
## 75%          26.435265       23.352309       21.08947      22.40036
##     Josh Brown Connor Barth Steven Hauschka Mason Crosby Cody Parkey
## 25%   0.000000      0.00000         0.00000      0.00000    0.000000
## 50%   2.051815      1.00000         1.00000      1.00000    1.977872
## 75%  22.587997     20.08152        20.97745     19.41744   19.652535
##     Adam Vinatieri Dan Bailey Matt Bryant Chandler Catanzaro Dan Carpenter
## 25%        0.00000    0.00000     0.00000              0.000       0.00000
## 50%        1.00000    0.00000     0.00000              0.000       0.00000
## 75%       19.54135   16.60393    16.73471             14.973      13.97517
##     Blair Walsh Graham Gano Caleb Sturgis Cairo Santos Phil Dawson
## 25%     0.00000     0.00000       0.00000      0.00000     0.00000
## 50%     0.00000     0.00000       0.00000      0.00000     0.00000
## 75%    10.58371    10.06175      11.08463     10.32753    12.02402
##     Nick Novak Drew Brees Andrew Luck Aaron Rodgers Russell Wilson
## 25%    0.00000   3.834008    4.273982       1.00000        0.00000
## 50%    0.00000  30.067149   26.993254      25.30654       13.04881
## 75%   10.29833  54.768182   54.085436      49.82085       38.02387
##     Ben Roethlisberger Peyton Manning Matt Ryan Eli Manning Cam Newton
## 25%            0.00000        0.00000  0.000000    0.000000     0.0000
## 50%           12.89020       13.09563  2.208871    5.021835     0.0000
## 75%           38.22426       37.49874 26.067544   29.525523    21.7968
##     Tony Romo Matthew Stafford Ryan Tannehill Philip Rivers
## 25%   0.00000           0.0000        0.00000       0.00000
## 50%   1.00000           0.0000        0.00000       0.00000
## 75%  19.43392          14.8006       17.37029      12.29987
##     Colin Kaepernick Joe Flacco Jay Cutler Teddy Bridgewater Carson Palmer
## 25%         0.000000   0.000000   0.000000          0.000000             0
## 50%         0.000000   0.000000   0.000000          0.000000             0
## 75%         7.968719   3.555617   5.346403          2.632346             1
##     Sam Bradford Alex Smith Le'Veon Bell Jamaal Charles Marshawn Lynch
## 25%            0          0     15.49119       15.23466       19.13207
## 50%            0          0     33.40082       32.94941       36.89198
## 75%            1          0     59.25476       59.95433       63.99235
##     Eddie Lacy Adrian Peterson Matt Forte C.J. Anderson DeMarco Murray
## 25%   14.49795        17.62661   1.198837       5.70098        1.00000
## 50%   30.69380        33.64728  18.123501      20.91135       16.94928
## 75%   54.85165        61.45452  44.844943      49.24171       43.95511
##     LeSean McCoy Jeremy Hill Mark Ingram Justin Forsett Lamar Miller
## 25%     1.359215    1.584332    2.319678       0.000000     0.000000
## 50%    17.306156   17.574647   16.900841       6.275818     6.247331
## 75%    46.506862   45.842328   42.510586      31.802509    34.850801
##     Alfred Morris Melvin Gordon Frank Gore Latavius Murray Carlos Hyde
## 25%      0.000000       0.00000    0.00000         0.00000     0.00000
## 50%      3.639537       1.00000    0.00000         0.00000     0.00000
## 75%     31.423620      27.81641   24.76082        23.84873    24.94708
##     Rashad Jennings Andre Ellington Joseph Randle T.J. Yeldon
## 25%         0.00000         0.00000        0.0000     0.00000
## 50%         0.00000         0.00000        0.0000     0.00000
## 75%        21.22932        13.95007       21.9186    16.14308
##     Jonathan Stewart Isaiah Crowell LeGarrette Blount Joique Bell
## 25%           0.0000        0.00000            0.0000     0.00000
## 50%           0.0000        0.00000            0.0000     0.00000
## 75%          17.0207       14.73267           18.9402    10.45793
##     Arian Foster Todd Gurley C.J. Spiller Christopher Ivory Tevin Coleman
## 25%     0.000000     0.00000            0           0.00000      0.000000
## 50%     0.000000     0.00000            0           0.00000      0.000000
## 75%     7.690317    11.38724            0          11.61935      5.957246
##     Giovani Bernard Ameer Abdullah Devonta Freeman Doug Martin
## 25%               0              0               0           0
## 50%               0              0               0           0
## 75%               1              1               1           1
##     Shane Vereen Ryan Mathews Bishop Sankey Alfred Blue Tre Mason
## 25%            0            0             0           0         0
## 50%            0            0             0           0         0
## 75%            0            1             0           0         0
##     Antonio Brown Odell Beckham Jr. Demaryius Thomas Dez Bryant
## 25%      24.48991          22.64237         16.84852   13.79191
## 50%      39.18147          35.97073         32.32182   28.94474
## 75%      55.65628          50.95575         47.86550   44.80823
##     Calvin Johnson Julio Jones Randall Cobb A.J. Green Alshon Jeffery
## 25%       8.320199   6.047803    9.313983   2.419343       1.975478
## 50%      23.371841  20.554507   24.235843  16.129324      17.122881
## 75%      40.540054  35.747709   40.019546  32.566252      33.240462
##     Mike Evans Brandin Cooks T.Y. Hilton Emmanuel Sanders DeAndre Hopkins
## 25%    0.00000       0.00000     0.00000          0.00000        0.000000
## 50%   12.27089      11.76539    12.55105         10.67651        3.135681
## 75%   29.63915      28.01381    28.73433         26.11080       19.816144
##     Jordan Matthews Julian Edelman DeSean Jackson Andre Johnson
## 25%        0.000000       0.000000        0.00000      0.000000
## 50%        4.181219       3.363046        2.16037      3.517004
## 75%       19.939486      19.249848       18.81803     17.632265
##     Golden Tate Keenan Allen Martavis Bryant Sammy Watkins Davante Adams
## 25%     0.00000      0.00000        0.000000      0.000000      0.000000
## 50%     1.00000      0.00000        3.589716      2.291064      2.157044
## 75%    16.46994     14.60557       19.891983     18.736579     16.950062
##     Jeremy Maclin Brandon Marshall Vincent Jackson Marques Colston
## 25%       0.00000         0.000000         0.00000         0.00000
## 50%       0.00000         3.083948         0.00000         0.00000
## 75%      13.28028        16.949354        12.32889        13.50042
##     Amari Cooper Mike Wallace Victor Cruz Jarvis Landry Eric Decker
## 25%      0.00000      0.00000     0.00000       0.00000      0.0000
## 50%      0.00000      0.00000     0.00000       0.00000      0.0000
## 75%     10.72261     11.81896    10.75676      10.83015      9.9835
##     Steve Smith Roddy White Anquan Boldin Allen Robinson Michael Floyd
## 25%      0.0000    0.000000      0.000000       0.000000       0.00000
## 50%      0.0000    0.000000      0.000000       0.000000       0.00000
## 75%     11.2896    9.993952      8.048359       9.889413      10.60002
##     Charles Johnson Nelson Agholor Brandon LaFell Torrey Smith
## 25%         0.00000        0.00000       0.000000      0.00000
## 50%         0.00000        0.00000       0.000000      0.00000
## 75%        10.71554        9.42703       9.273525     11.22739
##     Larry Fitzgerald John Brown Devin Funchess Rueben Randle Pierre Garcon
## 25%         0.000000   0.000000       0.000000       0.00000      0.000000
## 50%         0.000000   0.000000       0.000000       0.00000      0.000000
## 75%         9.434061   8.728283       6.895078       7.16263      5.905979
##     Kendall Wright Terrance Williams Dwayne Bowe Malcom Floyd Doug Baldwin
## 25%       0.000000          0.000000    0.000000     0.000000     0.000000
## 50%       0.000000          0.000000    0.000000     0.000000     0.000000
## 75%       4.767724          9.549511    4.878963     4.325862     7.451019
##     Breshad Perriman Eddie Royal Brian Quick Kenny Stills Kenny Britt
## 25%         0.000000    0.000000    0.000000     0.000000           0
## 50%         0.000000    0.000000    0.000000     0.000000           0
## 75%         5.582178    4.632805    2.565906     2.237647           1
##     Percy Harvin Michael Crabtree Steve Johnson Devante Parker
## 25%     0.000000                0             0              0
## 50%     0.000000                0             0              0
## 75%     3.307635                0             0              0
##     Rob Gronkowski Jimmy Graham Greg Olsen Travis Kelce Martellus Bennett
## 25%       20.47484     8.564926   0.000000     1.000000           0.00000
## 50%       27.97967    16.009881   6.441023     6.911866           3.45613
## 75%       40.07008    27.032943  19.887725    19.137537          15.02285
##     Jason Witten Julius Thomas Zach Ertz Delanie Walker Heath Miller
## 25%     0.000000       0.00000   0.00000       0.000000     0.000000
## 50%     4.258461       1.00000   0.00000       0.000000     0.000000
## 75%    16.969194      10.64255   7.22369       7.414059     4.454446
##     Jordan Cameron Dwayne Allen Kyle Rudolph Coby Fleener Owen Daniels
## 25%       0.000000       0.0000      0.00000      0.00000     0.000000
## 50%       0.000000       1.0000      0.00000      0.00000     0.000000
## 75%       4.575364      11.9798     10.23836     10.64484     5.038059
##     Larry Donnell Antonio Gates Jordan Reed Tyler Eifert
## 25%      0.000000       0.00000    0.000000     0.000000
## 50%      0.000000       0.00000    0.000000     0.000000
## 75%      4.406754      12.36738    2.698182     1.002373
##     Austin Seferian-Jenkins
## 25%                0.000000
## 50%                0.000000
## 75%                1.288861
```

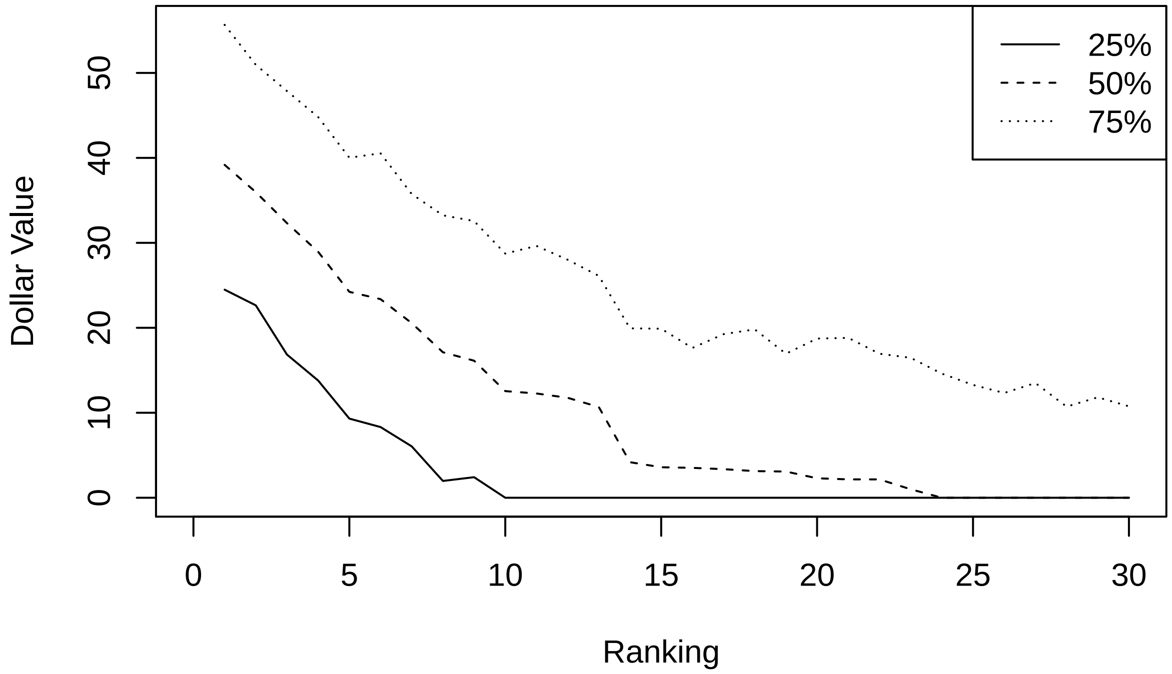
```
ci <- conf.interval(l1)
plot(ci, 'qb')
```



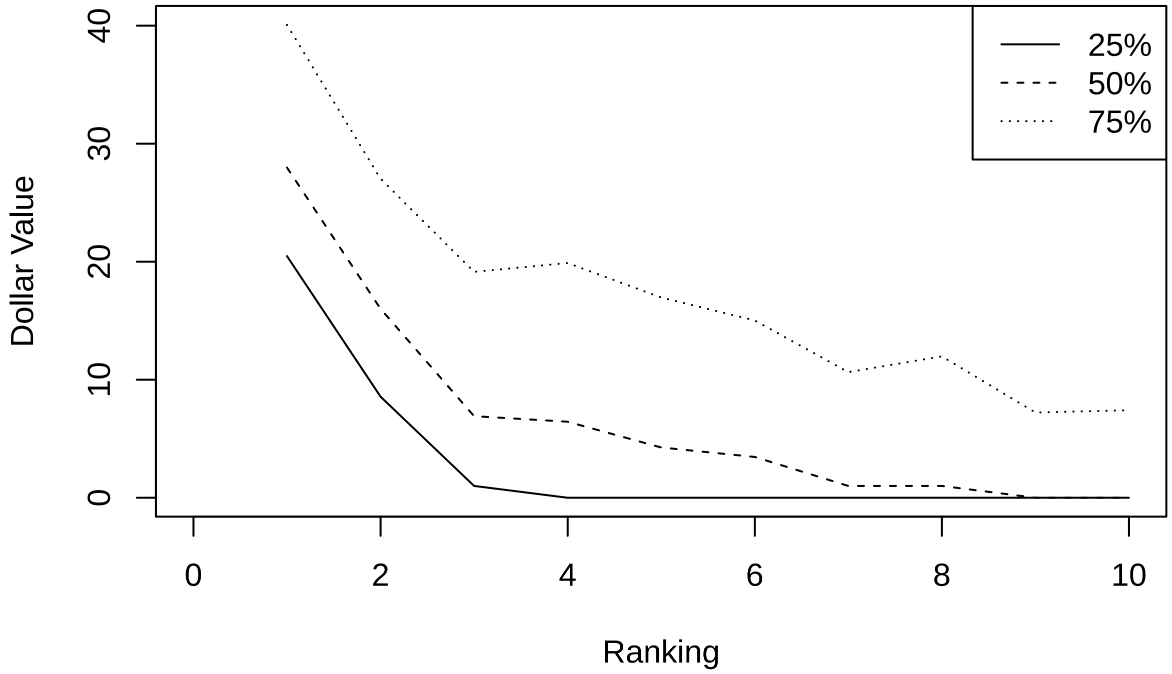
```
plot(ci, 'rb')
```



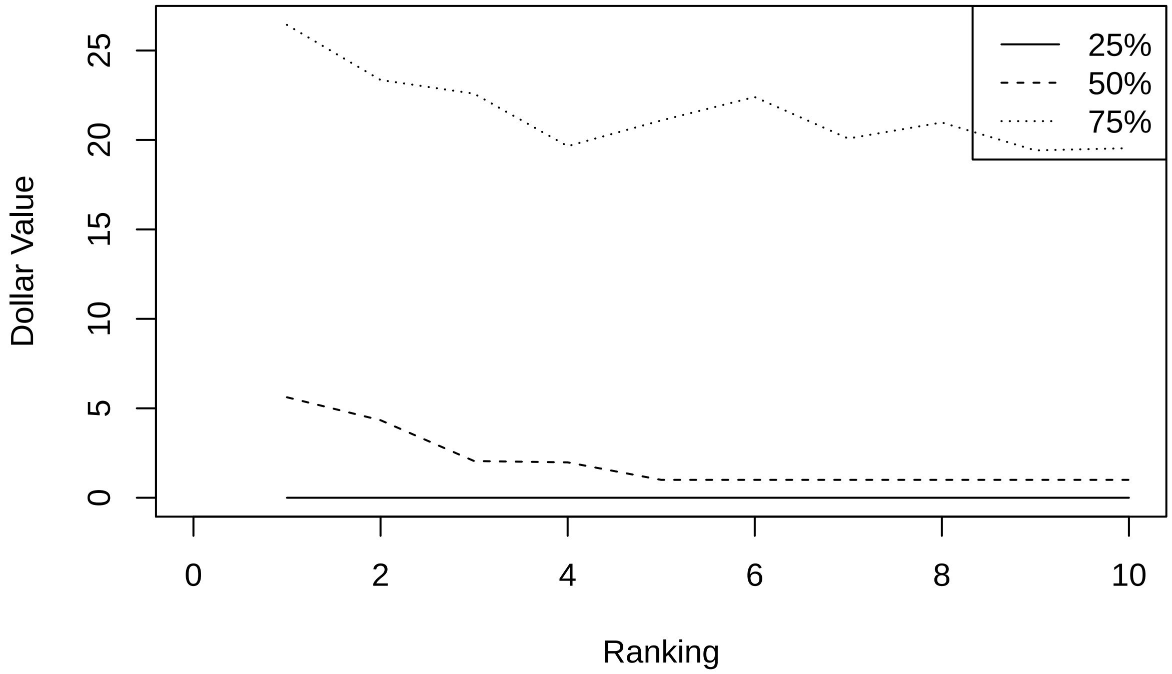
```
plot(ci, 'wr')
```



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



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


25%
50%
75%
Dollar Value
0
5
10
15
20
25
Ranking
0
2
4
6
8
10