Bios 301: Final Project

Utsav Kumar

12/09/2014

Due Tuesday, 9 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 CSV files in your working directory (note the new file `nfl_current14.csv`). You may utilize assignment 2, question 3 in your solution.

Task 1: Finding Residuals (80 points)

At the beginning of the course we examined projections for the 2014 NFL season. With the season ~65% completed, let's compare the observed values to the estimated values. Place all code at the end of the instructions.

1. Read in the five projection files
2. Add a column for position
3. Combine these files
4. The projection data assumes teams play 16 games. The observed data is taken after week 11, where 30 teams have played 10 games, and 2 teams (`PIT` and `CAR`) have played 11. The projection data includes a column for a player's team. In the projection data, multiply the numeric columns by percent of the season played (so 11/16 if team is PIT/CAR, otherwise, 10/16).
5. Read in the observed data (`nfl_current14.csv`)
6. Merge the projected data with the observed data by the player's name
7. Determine the proper mapping of categories in the projected data to the observed data. For instance, fg=>FGM, fga=>FGA, xpt=>XPM (the rest should be easy). Hint: there should be 15 mapped categories.
8. Create the following vector `pred.pos <- c(k=20, qb=20, rb=40, wr=60, te=20)`. This is the number of players to examine at each position.
9. Create a list of residual values by position
10. When subsetting by position, re-order the data as follows:

- if position = 'k', order by projected 'fg' descendingly
- if position = 'qb', order by projected 'pass_tds' descendingly
- if position = 'rb', order by projected 'rush_tds' descendingly
- if position = 'wr', order by projected 'rec_tds' descendingly
- if position = 'te', order by projected 'rec_tds' descendingly

1. Reduce the subset to the number of rows given above. For instance, look at the top 40 'rb', ordered by projected rushing touchdowns.
2. Take the difference between the observed data and the projected data for each category. Thus, the residuals for 'rb' should be a 40x15 data.frame or matrix.

Stub Answer The result of task 1 is necessary to complete task 3. If you were unable to generate a list of residuals, please use the following code to simulate residuals. Otherwise, you do not need this code.

```
pred.pos <- c(k=20, qb=20, rb=40, wr=60, te=20)
noise <- list()
empty <- data.frame(fg=0, fga=0, xpt=0, pass_att=0, pass_cmp=0, pass_yds=0, pass_tds=0, pass_ints=0,
                    rush_att=0, rush_yds=0, rush_tds=0, fumbles=0, rec_att=0, rec_yds=0, rec_tds=0)
for(i in names(pred.pos)) {
  noise[[i]] <- empty[rep(1,pred.pos[i]),]
  row.names(noise[[i]]) <- NULL
}
set.seed(35)
noise$k$fg <- rnorm(20, 0, 3)
noise$k$fga <- rnorm(20, 0, 4)
noise$k$xpt <- rnorm(20, 0, 5)
noise$qb$pass_att <- rnorm(20, 0, 25)
noise$qb$pass_cmp <- rnorm(20, 0, 15)
noise$qb$pass_yds <- rnorm(20, 0, 150)
noise$qb$pass_tds <- rnorm(20, 0, 4)
noise$qb$pass_ints <- rnorm(20, 0, 2)
noise$rb$rush_att <- rnorm(40, 0, 40)
noise$rb$rush_yds <- rnorm(40, 0, 200)
noise$rb$rush_tds <- rnorm(40, 0, 2)
noise$rb$fumbles <- rnorm(40, 0, 1)
noise$wr$rec_att <- rnorm(60, 0, 10)
noise$wr$rec_yds <- rnorm(60, 0, 160)
noise$wr$rec_tds <- rnorm(60, 0, 2)
noise$te$rec_att <- rnorm(20, 0, 10)
noise$te$rec_yds <- rnorm(20, 0, 120)
noise$te$rec_tds <- rnorm(20, 0, 2)
```

```
# PART 1

# Reading data files for projected data
  k <- read.csv('proj_k14.csv', header=TRUE, stringsAsFactors=FALSE)
  qb <- read.csv('proj_qb14.csv', header=TRUE, stringsAsFactors=FALSE)
  rb <- read.csv('proj_rb14.csv', header=TRUE, stringsAsFactors=FALSE)
  te <- read.csv('proj_te14.csv', header=TRUE, stringsAsFactors=FALSE)
  wr <- read.csv('proj_wr14.csv', header=TRUE, stringsAsFactors=FALSE)

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

# create a new column in each data.frame with playing position type
  k[,'pos'] <- 'k'
  qb[,'pos'] <- 'qb'
  rb[,'pos'] <- 'rb'
  te[,'pos'] <- 'te'
  wr[,'pos'] <- 'wr'

# append 'pos' to unique column list
  cols <- c(cols, 'pos')

# create common columns in each data.frame
```

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

# rename column names, by removing periods
  names(x) <- gsub('[.]', '', names(x))

# data.frame stored in stat

  stat <- x

# reading nfl current data
 nfl <- read.csv("nfl_current14.csv", header = TRUE, stringsAsFactors = FALSE)
 names(nfl) <- c("PlayerName", "XTeam", "fga", "fg", "xpt", "pass_cmp", "pass_att",
                 "pass_yds", "pass_tds", "pass_ints", "rec_att", "rec_yds",
                 "rec_tds", "rush_att", "rush_yds", "rush_tds", "fumbles")

# multiplying the data on the basis of percent completion

  col_mod <- c("fg","fga","xpt","fpts","pass_att","pass_cmp","pass_yds","pass_tds",
               "pass_ints","rush_att","rush_yds","rush_tds","fumbles","rec_att",
               "rec_yds","rec_tds")


# 11/16 for PIT and CAR and 10/16 for rest
  x[x$XTeam == "PIT" || x$XTeam == "CAR", col_mod] <-
  x[x$XTeam == "PIT" || x$XTeam == "CAR", col_mod] * 11/16

  x[x$XTeam != "PIT" && x$XTeam !="CAR", col_mod] <-
  x[x$XTeam != "PIT" && x$XTeam != "CAR", col_mod] * 10/16

# Storing the projected data after introducing % game completion into stat
  stats <- x

# merging the two data files on the basis of names
# After merging .x represent the projected data
# .y represent the nfl stats
  merge.x <- merge(x, nfl, by.x = "PlayerName", by.y = "PlayerName", all = FALSE)

# No of players examine at each position
  pred.pos <- c(k=20, qb=20, rb=40, wr=60, te=20)

# ordering the player in order of their play position
  merge.x <- merge.x[order(merge.x$pos),]

# subsetting the data to extract the players
  k.x <- subset(merge.x, pos == "k")
```

```
  qb.x <- subset(merge.x, pos == "qb")
  rb.x <- subset(merge.x, pos == "rb")
  te.x <- subset(merge.x, pos == "te")
  wr.x <- subset(merge.x, pos == "wr")

# ordering the players in decreasing order of given parameters
  k.x <- k.x[order(-k.x[,"fg.x"]),]
  qb.x <- qb.x[order(-qb.x[,"pass_tds.x"]),]
  rb.x <- rb.x[order(-rb.x[,"rush_tds.x"]),]
  te.x <- te.x[order(-te.x[,"rec_tds.x"]),]
  wr.x <- wr.x[order(-wr.x[,"rec_tds.x"]),]

# Subsetting the data according to the given numbers in list pred.pos
  k.x <- k.x[1:pred.pos[["k"]],]
  qb.x <- qb.x[1:pred.pos[["qb"]],]
  rb.x <- rb.x[1:pred.pos[["rb"]],]
  te.x <- te.x[1:pred.pos[["te"]],]
  wr.x <- wr.x[1:pred.pos[["wr"]],]

# finding residuals for different play position
  col.names <- c("fga", "fg", "xpt", "pass_cmp", "pass_att","pass_yds", "pass_tds",
                 "pass_ints", "rec_att", "rec_yds", "rec_tds", "rush_att",
                 "rush_yds", "rush_tds", "fumbles")
# creating data fram for residuals
  residuals <- rbind(k.x, qb.x, rb.x, te.x, wr.x)
  residuals[,col.names] <- 0
  residuals["fga"] <- residuals["fga.x"] - residuals["fga.y"]
  residuals["fg"] <- residuals["fg.x"] - residuals["fg.y"]
  residuals["xpt"] <- residuals["xpt.x"] - residuals["xpt.y"]
  residuals["pass_cmp"] <- residuals["pass_cmp.x"] - residuals["pass_cmp.y"]
  residuals["pass_att"] <- residuals["pass_att.x"] - residuals["pass_att.y"]
  residuals["pass_yds"] <- residuals["pass_yds.x"] - residuals["pass_yds.y"]
  residuals["pass_tds"] <- residuals["pass_tds.x"] - residuals["pass_tds.y"]
  residuals["pass_ints"] <- residuals["pass_ints.x"] - residuals["pass_ints.y"]
  residuals["rec_att"] <- residuals["rec_att.x"] - residuals["rec_att.y"]
  residuals["rec_yds"] <- residuals["rec_yds.x"] - residuals["rec_yds.y"]
  residuals["rec_tds"] <- residuals["rec_tds.x"] - residuals["rec_tds.y"]
  residuals["rush_att"] <- residuals["rush_att.x"] - residuals["rush_att.y"]
  residuals["rush_yds"] <- residuals["rush_yds.x"] - residuals["rush_yds.y"]
  residuals["rush_tds"] <- residuals["rush_tds.x"] - residuals["rush_tds.y"]
  residuals["fumbles"] <- residuals["fumbles.x"] - residuals["fumbles.y"]

# subsetting residuals for different position ordered on the basis of given parameter
# for each like projected fg for k, projected pass_tds for qb etc
  residuals.k <- subset(residuals[,col.names], residuals$pos == "k")
  residuals.qb <- subset(residuals[,col.names], residuals$pos == "qb")
  residuals.rb <- subset(residuals[,col.names], residuals$pos == "rb")
  residuals.te <- subset(residuals[,col.names], residuals$pos == "te")
  residuals.wr <- subset(residuals[,col.names], residuals$pos == "wr")

# Forming a list of residual on the basis of play position
residual.list <- list(residuals.k = residuals.k, residuals.qb = residuals.qb,
                      residuals.rb = residuals.rb, residuals.te = residuals.te,
```

```
                        residuals.wr = residuals.wr)
```

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`, `pos`, `points`)

- Inside the function, save the `stats` argument as an element of a list
- The other arguments can also be saved on the list, or saved as attributes of the list
- Define the class of the list as `league`
- Add another list element, `fan_pnts`, which is assigned the output of function `calcPoints` (defined below)
- Add another list element, `values`, which is assigned the output of function `buildValues` (defined below)
- Add another list element, `sims`, and assign it to NULL
- Return the list (which should be a S3 class now)
- It's worth mentioning that in OOP terminology, `stats`, `fan_pnts`, `values`, and `sims` are attributes of the `league` class. This may be confusing as any R variable may have attributes (see ?attr or ?attributes). Henceforth, I will simply refer to `stats`, etc as list elements. `nteams`, `cap`, `pos`, and `points` will either be list elements or list attributes (depending on how you define your class), and I will refer to them as settings.

1. Create a function `calcPoints` that takes 1 argument, a league object

- Use the `stats` list element and the `points` setting from the league object
- Calculate points for each stat category by multiplying that stat by its respective point multiplier
- Create a new column `Points`, which is the sum of all the points categories
- Order the data set by `Points` descendingly
- Return the data set with the following columns: `PlayerName`, `XTeam`, `pos`, `Points`
- Note this will be assigned to the `fan_pnts` list element

1. Create a function `buildValues` that takes 1 argument, a league object

- Use the `fan_pnts` list element and the `nteams`, `cap`, and `pos` settings
- Order the data set by `Points` descendingly
- Create a new column `marg`
- Order the data set by `marg` descendingly
- Create a new column `value`
- Return the data set with the following columns: `PlayerName`, `XTeam`, `pos`, `Points`, `marg`, `value`
- Note this will be assigned to the `values` list element

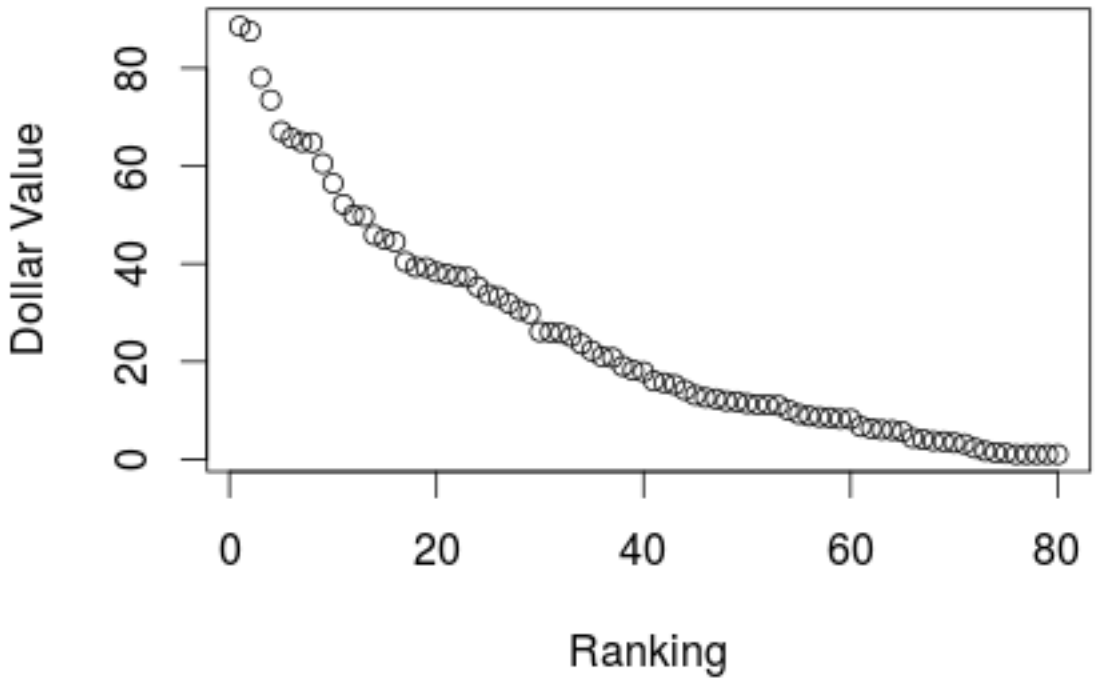
1. Create a `print` method for the league class

- Print the `values` list element

1. Create a `plot` method for the league class

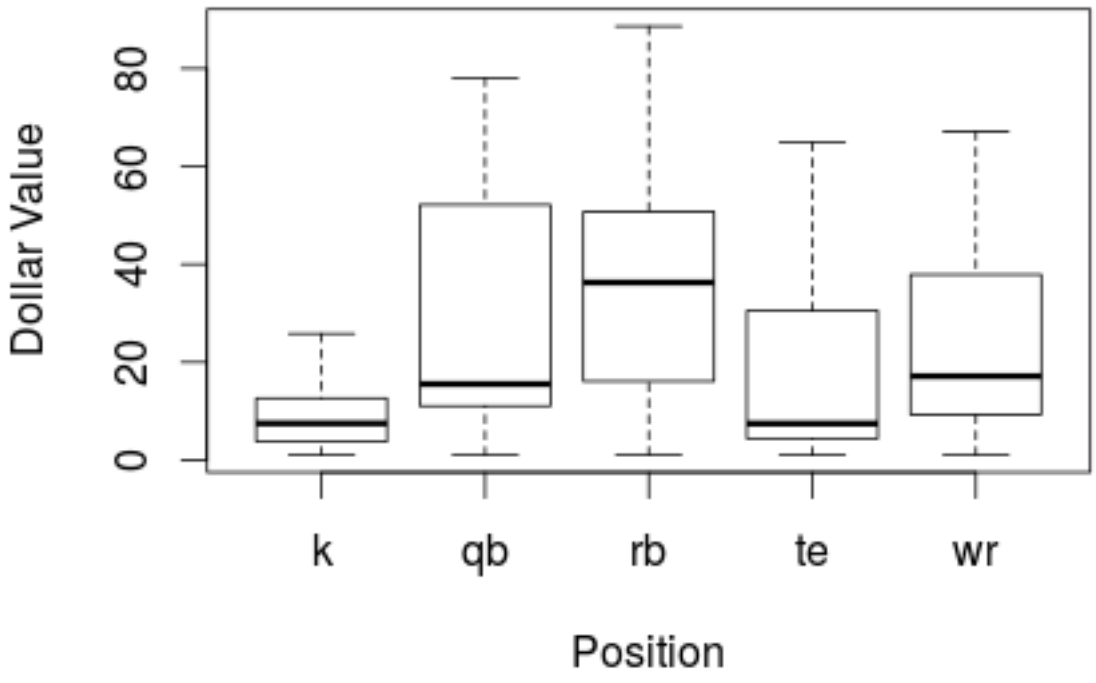
- Plot the `value` column of the `values` list element

- Add minimal plotting decorations (such as axis labels)
- Here's an example:



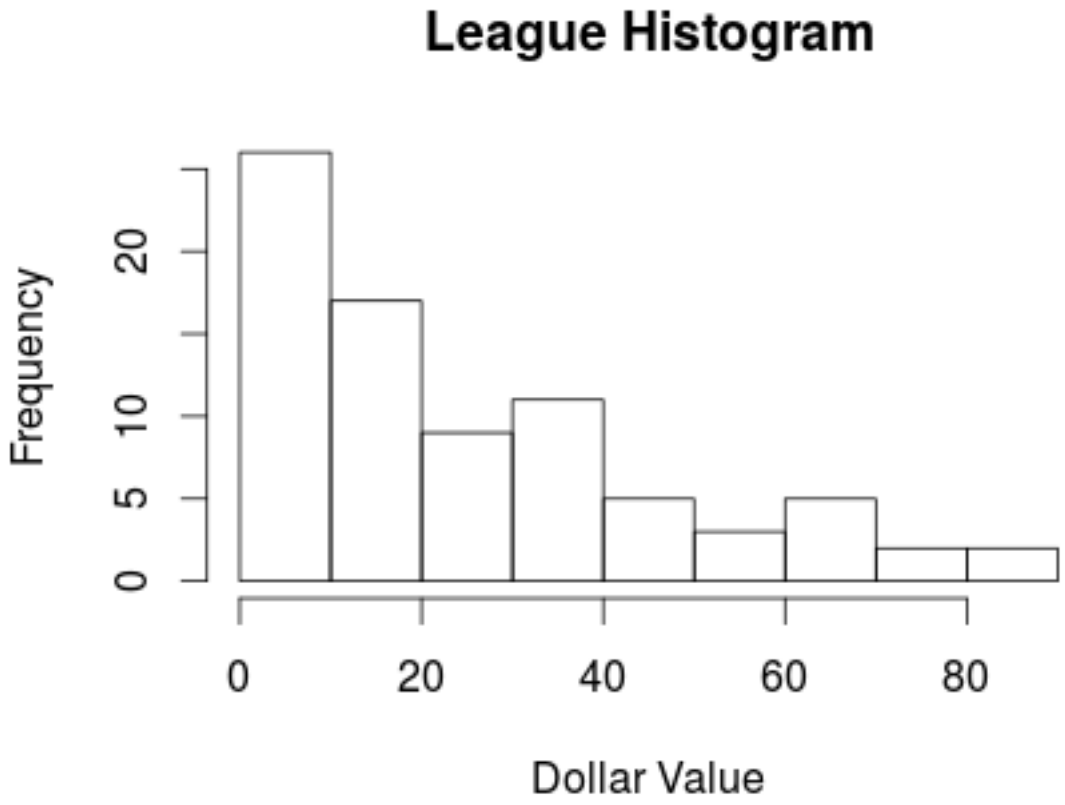
1. Create a `boxplot` method for the league class

- Run `boxplot` of `value` by `pos` from the `values` list element
- Add minimal plotting decorations
- Here's an example:



1. Create a `hist` method for the league class

- Run `hist` on the `value` column of the `values` list element
- Add minimal plotting decorations
- Here's an example:



I will test your code with the following:

```
r # 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(x, 10, 200, pos, 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`.

```
# Defining class league
league <- function(stats, nteams, cap, pos, points)
{
  x <- list(stats = stats, nteams = nteams, cap = cap, pos = pos, points = points)
  fan_pts <- calcPoints(x)
  x <- list(stats = stats, nteams = nteams, cap = cap, pos = pos, points = points,
```

```
            fan_pts = fan_pts)
  values <- buildValues(x)
  x <- list(stats = stats, nteams = nteams, cap = cap, pos = pos, points = points,
            fan_pts = fan_pts, values = values)
  x <- list(stats = stats, nteams = nteams, cap = cap, pos = pos, points = points,
            fan_pts = fan_pts, values = values, sims = NULL)
  class(x) <- "league"
  return(x)
}

# function calcPoints
calcPoints <- function(league.object)
{
# reading stats and points
  x <- league.object[["stats"]]
  points <- league.object[["points"]]

# points supplied as a parameter to the function right now assumed as variable
  fg = as.numeric(points["fg"])
  xpt = as.numeric(points["xpt"])
  pass_yds = as.numeric(points["pass_yds"])
  pass_tds = as.numeric(points["pass_tds"])
  pass_ints = as.numeric(points["pass_ints"])
  rush_yds = as.numeric(points["rush_yds"])
  rush_tds = as.numeric(points["rush_tds"])
  fumbles = as.numeric(points["fumbles"])
  rec_yds = as.numeric(points["rec_yds"])
  rec_tds = as.numeric(points["rec_tds"])

# convert NFL stat to fantasy points
  x[,'p_fg'] <- x[,'fg']*fg
  x[,'p_xpt'] <- x[,'xpt']*xpt
  x[,'p_pass_yds'] <- x[,'pass_yds']*pass_yds
  x[,'p_pass_tds'] <- x[,'pass_tds']*pass_tds
  x[,'p_pass_ints'] <- x[,'pass_ints']*pass_ints
  x[,'p_rush_yds'] <- x[,'rush_yds']*rush_yds
  x[,'p_rush_tds'] <- x[,'rush_tds']*rush_tds
  x[,'p_fumbles'] <- x[,'fumbles']*fumbles
  x[,'p_rec_yds'] <- x[,'rec_yds']*rec_yds
  x[,'p_rec_tds'] <- x[,'rec_tds']*rec_tds

# summing the points along the row
  x[,'Points'] <- rowSums(x[,grep("^p_", names(x))])

# setting up the points column in decreasing order
  x2 <- x[order(x[,'Points'], decreasing=TRUE),]

# forming and returning final dataframe
  finalx <- subset(x2, select = c(PlayerName, XTeam, pos, Points))

  return(finalx)
}
```

```
# function calculate buildvalues
buildValues <- function(league.object)
{

# reading fan_pts and Points
  x <- league.object[["fan_pts"]]
  x2 <- x[order(x[,'Points'], decreasing=TRUE),]

# determine the row indices for each position
  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')

# no. of player in the team required given as parameter to function
  posReq <- league.object[["pos"]]
  qb = as.numeric(posReq["qb"])
  rb = as.numeric(posReq["rb"])
  wr = as.numeric(posReq["wr"])
  te = as.numeric(posReq["te"])
  k = as.numeric(posReq["k"])

# calculate marginal points by subtracting "baseline" player's points
# if else statement to take care of the no. of player for a position
# required is 0

  nteams <- league.object[["nteams"]]
  if (k > 0)
  {x2[k.ix, 'marg'] <- x2[k.ix,'Points'] - x2[k.ix[nteams*k],'Points']}
  else
  {x2[k.ix, 'marg'] <- -1}
  if (qb > 0)
  {x2[qb.ix, 'marg'] <- x2[qb.ix,'Points'] - x2[qb.ix[nteams*qb],'Points']}
  else
  {x2[qb.ix, 'marg'] <- -1}
  if (rb > 0)
  {x2[rb.ix, 'marg'] <- x2[rb.ix,'Points'] - x2[rb.ix[nteams*rb],'Points']}
  else
  {x2[rb.ix, 'marg'] <- -1}
  if (te > 0)
  {x2[te.ix, 'marg'] <- x2[te.ix,'Points'] - x2[te.ix[nteams*te],'Points']}
  else
  {x2[te.ix, 'marg'] <- -1}
  if (wr > 0)
  {x2[wr.ix, 'marg'] <- x2[wr.ix,'Points'] - x2[wr.ix[nteams*wr],'Points']}
  else
  {x2[wr.ix, 'marg'] <- -1}

# create a new data.frame subset by non-negative marginal points
  x3 <- x2[x2[,'marg'] >= 0,]

# re-order by marginal points
```

```
  x3 <- x3[order(x3[,'marg'], decreasing=TRUE),]

# reset the row names
  rownames(x3) <- NULL

# calculation for player value
  x3[,'value'] <- x3[,'marg']*(nteams*cap-nrow(x3))/sum(x3[,'marg']) + 1

# create a data.frame with required columns needed in data.frame
  player.each.team <- qb + rb + wr + te + k
  x4 <- x3[1:(nteams*player.each.team), c('PlayerName', 'XTeam', 'pos',
                                          'Points', 'marg' ,'value')]

# returning final data.frame
  return(x4)
}
```

```
# Running the league function
# stats have been stored and described in PART 1
# stats: I have used the projected data with the % game completion
# stat: Without considering % game completion
# Here stats has the data for the projected case
# Other parameters that needs to be send to function league
# storing stats into x

# please use stats instead of x for stats part since I have used it
# many times as variable during my simulation
nteams <- 10
cap <- 200
pos <- list(qb=1, rb=2, wr=3, te=1, k=1)
points <- 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)

# league.object storing object of class league with given data elements
league.object <- league(stats, nteams, cap, pos, points)
```

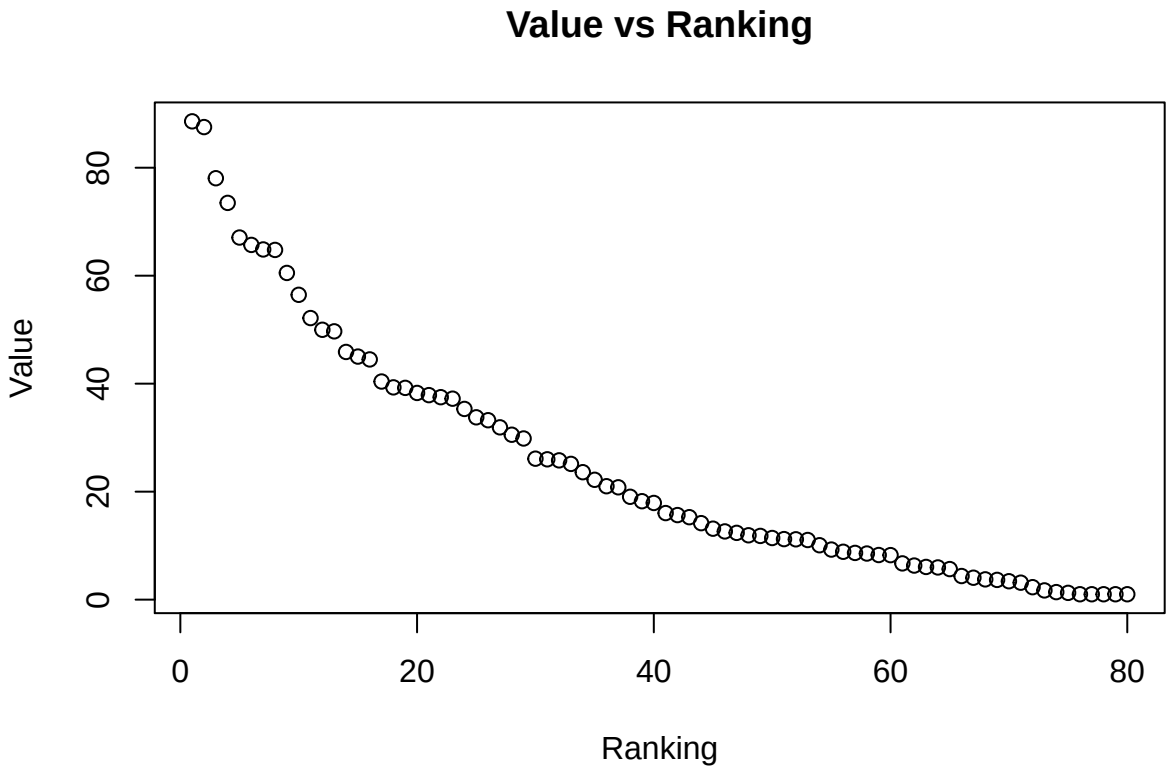
```
# Print Class
print.league <- function(league.object, class = "league")
{
  values <- league.object[["values"]]
  print(values)
}

print(league.object)
```

```
##              PlayerName XTeam pos Points    marg  value
## 1         Jamaal Charles    KC  rb 148.11 53.6437 88.585
## 2           LeSean McCoy   PHI  rb 147.46 52.9937 87.524
## 3         Peyton Manning   DEN  qb 226.50 47.1912 78.050
## 4        Adrian Peterson   MIN  rb 138.86 44.3937 73.482
## 5         Calvin Johnson   DET  wr  93.46 40.4656 67.069
## 6             Matt Forte   CHI  rb 134.09 39.6281 65.701
## 7           Jimmy Graham    NO  te  85.26 39.1125 64.860
```

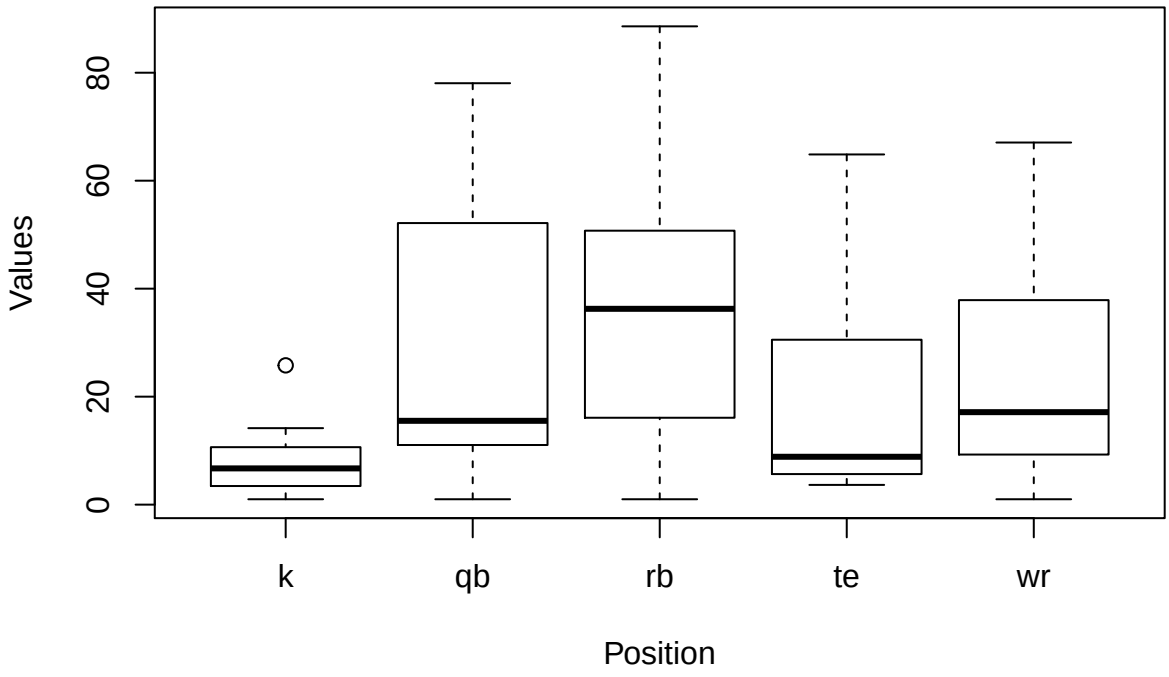
```
## 8       Demaryius Thomas   DEN  wr  92.05 39.0594 64.773
## 9          Aaron Rodgers    GB  qb 215.76 36.4488 60.510
## 10            Eddie Lacy    GB  rb 128.43 33.9656 56.456
## 11            Drew Brees    NO  qb 210.64 31.3300 52.153
## 12           Julio Jones   ATL  wr  82.99 30.0000 49.981
## 13            Dez Bryant   DAL  wr  82.81 29.8219 49.691
## 14            A.J. Green   CIN  wr  80.47 27.4844 45.874
## 15        Marshawn Lynch   SEA  rb 121.43 26.9625 45.022
## 16      Brandon Marshall   CHI  wr  79.63 26.6375 44.491
## 17           Montee Ball   DEN  rb 118.59 24.1281 40.394
## 18          Jordy Nelson    GB  wr  76.46 23.4656 39.313
## 19         Julius Thomas   DEN  te  69.56 23.4094 39.221
## 20        DeMarco Murray   DAL  rb 117.30 22.8344 38.282
## 21        Alshon Jeffery   CHI  wr  75.58 22.5875 37.879
## 22         Alfred Morris   WAS  rb 116.82 22.3531 37.496
## 23          Le'Veon Bell   PIT  rb 116.65 22.1844 37.221
## 24           Doug Martin    TB  rb 115.48 21.0187 35.318
## 25          Arian Foster   HOU  rb 114.53 20.0687 33.767
## 26         Antonio Brown   PIT  wr  72.73 19.7406 33.231
## 27             Zac Stacy   STL  rb 113.40 18.9312 31.909
## 28        Rob Gronkowski    NE  te  64.24 18.0937 30.542
## 29          Randall Cobb    GB  wr  70.66 17.6688 29.848
## 30       Giovani Bernard   CIN  rb 109.84 15.3781 26.108
## 31          Vernon Davis    SF  te  61.45 15.3062 25.991
## 32    Stephen Gostkowski    NE   k 112.94 15.1875 25.797
## 33      Larry Fitzgerald   ARI  wr  67.78 14.7844 25.139
## 34       Vincent Jackson    TB  wr  66.83 13.8437 23.603
## 35      Colin Kaepernick    SF  qb 192.29 12.9812 22.195
## 36          Toby Gerhart   JAC  rb 106.72 12.2531 21.006
## 37 Cordarrelle Patterson   MIN  wr  65.12 12.1281 20.802
## 38      Michael Crabtree    SF  wr  64.05 11.0562 19.052
## 39          Keenan Allen    SD  wr  63.54 10.5531 18.230
## 40      Matthew Stafford   DET  qb 189.67 10.3550 17.907
## 41            Wes Welker   DEN  wr  62.20  9.2063 16.031
## 42         Andre Johnson   HOU  wr  61.97  8.9813 15.664
## 43          Percy Harvin   SEA  wr  61.73  8.7406 15.271
## 44           Phil Dawson    SF   k 105.81  8.0625 14.164
## 45           Nick Foles   PHI  qb 186.75  7.4387 13.145
## 46         Justin Tucker   BAL   k 104.88  7.1250 12.633
## 47         Pierre Garcon   WAS  wr  59.95  6.9625 12.368
## 48           Roddy White   ATL  wr  59.68  6.6875 11.919
## 49         Michael Floyd   ARI  wr  59.61  6.6156 11.801
## 50           Victor Cruz   NYG  wr  59.37  6.3781 11.414
## 51           Andrew Luck   IND  qb 185.57  6.2550 11.213
## 52         Bishop Sankey   TEN  rb 100.70  6.2313 11.174
## 53    Robert Griffin III   WAS  qb 185.47  6.1513 11.043
## 54          Ryan Mathews    SD  rb 100.03  5.5594 10.077
## 55        DeSean Jackson   WAS  wr  58.06  5.0688  9.276
## 56          Jason Witten   DAL  te  50.97  4.8188  8.868
## 57       Steven Hauschka   SEA   k 102.44  4.6875  8.653
## 58            Cam Newton   CAR  qb 183.93  4.6188  8.541
## 59            Frank Gore    SF  rb  98.92  4.4500  8.266
## 60        Adam Vinatieri   IND   k 102.19  4.4375  8.245
## 61          Mason Crosby    GB   k 101.25  3.5000  6.715
```

```
## 62       Andre Ellington   ARI  rb  97.72  3.2531  6.311
## 63          Dennis Pitta   BAL  te  49.24  3.0969  6.056
## 64          Torrey Smith   BAL  wr  56.03  3.0406  5.964
## 65        Jordan Cameron   CLE  te  49.00  2.8531  5.658
## 66           Jordan Reed   WAS  te  48.21  2.0656  4.373
## 67            Dan Bailey   DAL   k  99.62  1.8750  4.061
## 68           Matt Prater   DEN   k  99.44  1.6875  3.755
## 69          Kyle Rudolph   MIN  te  47.77  1.6250  3.653
## 70         Jeremy Maclin   PHI  wr  54.47  1.4750  3.408
## 71            Nick Novak    SD   k  99.06  1.3125  3.143
## 72          Mike Wallace   MIA  wr  53.79  0.8000  2.306
## 73             Ty Hilton   IND  wr  53.42  0.4344  1.709
## 74      Emmanuel Sanders   DEN  wr  53.23  0.2406  1.393
## 75        Julian Edelman    NE  wr  53.15  0.1562  1.255
## 76             Matt Ryan   ATL  qb 179.31  0.0000  1.000
## 77         Shayne Graham    NO   k  97.75  0.0000  1.000
## 78           Matt Bryant   ATL   k  97.75  0.0000  1.000
## 79          C.J. Spiller   BUF  rb  94.47  0.0000  1.000
## 80       Marques Colston    NO  wr  52.99  0.0000  1.000
```

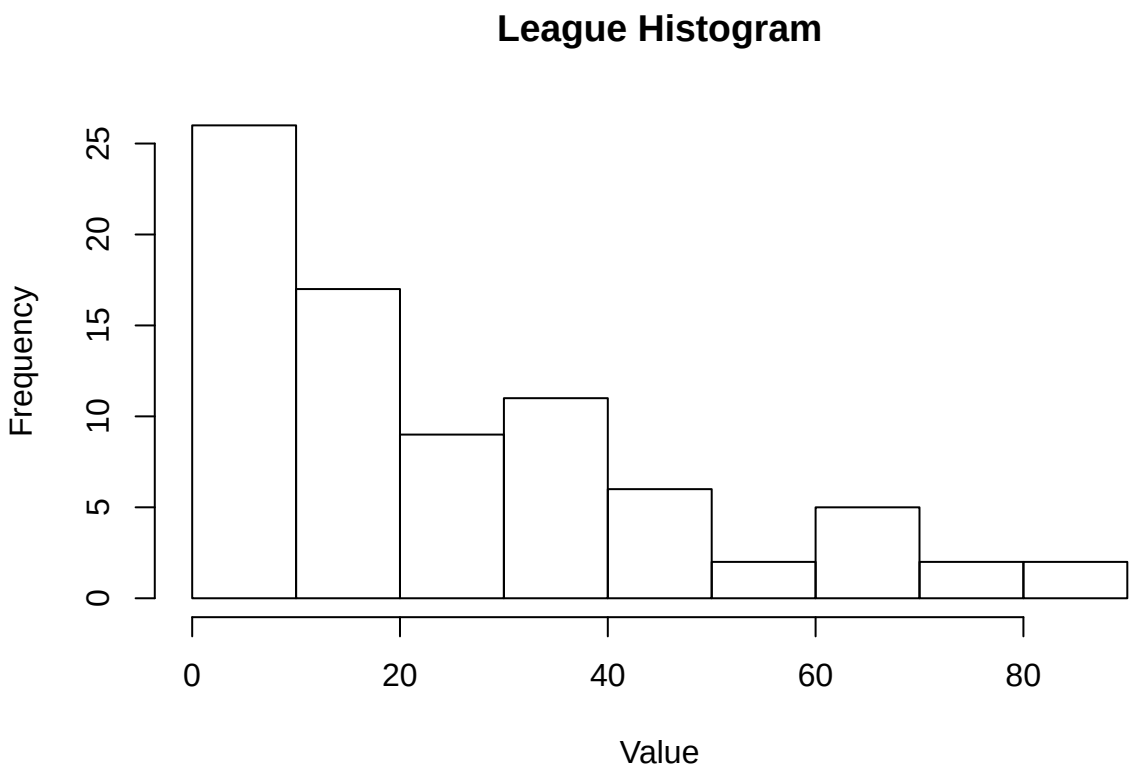
```r
# plotting scatter
plot.league <- function(league.object, class = "league")
{
  values <- league.object[["values"]]
  x <- seq(1:nrow(values))
  y <- values$value
  plot(x, y, main = "Value vs Ranking", xlab = "Ranking", ylab = "Value", pch = 1)
}

plot(league.object)
```




```r
# ploting boxplot
boxplot.league <- function(league.object, class = "league")
{
  x <- league.object[["values"]]
  pos <- x$pos
  value <- x$value
  boxplot(value~pos, main = "Value vs. Position", xlab = "Position", ylab = "Values")
}

boxplot(league.object)
```




```
# plotting histogram
hist.league <- function(league.object, class = "league")
{
  x <- league.object[["values"]]
  value <- x$value
  hist(value, main = "League Histogram", xlab = "Value", ylab = "Frequency")
}

hist(league.object)
```



Task 3: Simulations with Residuals (40 points)

Using residuals from task 1, create a list of league simulations to save in the list element `sims`. 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

- Use the `stats` list element and the `pos` setting
- Set a RNG seed to ensure reproducability
- For each simulation iterate over each position in the residuals
- Simulate new stats for the given position
- Assume residuals is a list, and the 'qb' element contains a 20x15 matrix
- Calculate the number of quarterbacks found in `stats` - assume this is 35 (which is not real value)
- Sample from 1 to 20 (with replacement) 35 times
- Use the sample of 35 to select rows from the residuals
- Add the 35 rows of residuals to the 35 rows of stats
- Note: stats can't be negative so replace any negative values with 0
- Use the new, simulated stats data set and the old league settings to generate a new league object
- Save the `values` list element of the new league object in a list for each simulation
- Merge the list of values by `PlayerName`
- Start with the `stats` data set, using the "PlayerName", "XTeam", "pos" columns
- `value` is the only column you need to merge from `values`
- Remember that the `values` list element only has rows for the number of required players while the `stats` list element has rows for all projected players. Thus some players will have values in some simulations, but not in others. When merging, set a player's value to 0 if he is not found in the `values` list element.

- If 1000 simulations are requested, you should end up with a data frame with 1003 columns
- Assign the merged values data set to the `sims` list element (of the original league object)
- Return the league object

1. Create a `quantile` method for the league class; it takes at least two arguments, a league object and a probs vector

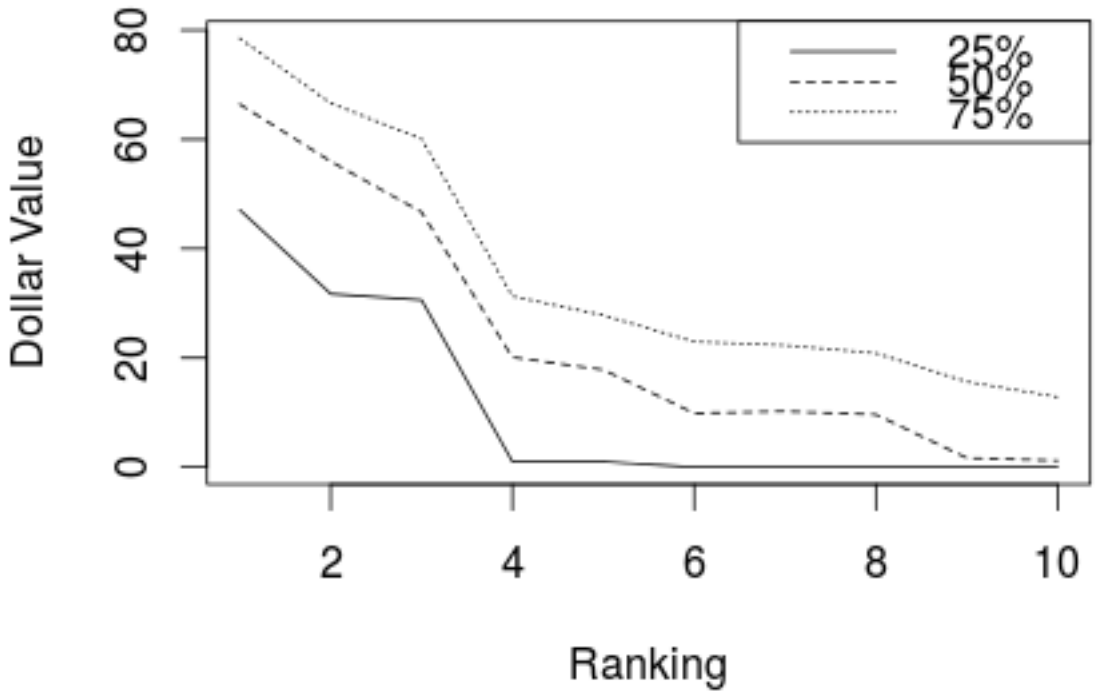
- This method requires the `sims` list element; it should fail if `sims` is NULL
- The `probs` vector should default to `c(0.25, 0.5, 0.75)`
- Run `quantile` for each row, passing it `probs` (and any additional arguments)
- Return a data set with columns for "PlayerName", "XTeam", "pos", and for each probability

1. Create a function `conf.interval`; it takes at least two arguments, a league object and a probs vector

- This method requires the `sims` list element; it should fail if `sims` is NULL
- Use the `nteams` and `pos` settings
- Call `quantile` on the league object
- Iterate over each position, saving results as a list
- Subset the output of quantile by position
- Order this data set by the last column (which should be the highest probability) descendingly
- Restrict the number of rows to the number of required players at this position
- Set the class of this list to `league.conf.interval`
- Return the new object

1. Create a `plot` method for the league.conf.interval class; it takes at least two arguments, a league.conf.interval object and a position

- Select the list element that matches the position - this data set holds the quantile values for each player
- Plot lines for each probability; using the defaults, you would have three lines (0.25, 0.5, 0.75)
- Add minimal plotting decorations (such as axis labels)
- Add 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')
```

```
# Function addNoise

# parameters of function league.object
# x <- list(stats = stats, nteams = nteams, cap = cap, pos = pos, points = points, fan_pts = fan_pts, v
# returned as league.object

# residual.list
# residual.list <- list(residuals.k, residuals.qb, residuals.rb, residuals.te, residuals.wr)

# Number of simulation nsims

# RNG seed rng.seed
addNoise <- function (league.object, residual.list, nsims, rng.seed)
```

```
{
# setting up the rng seed
  set.seed(rng.seed)
  sims <- data.frame(matrix(ncol = nsims + 3, nrow = 0))
  sim <- data.frame(matrix(ncol = 3, nrow = 0))
# Obtaining the stats from league class object
   x <- league.object[["stats"]]

# sims containing name, team and position for all the players
   sims[1:nrow(x),1:3] <- x[1:nrow(x),c('PlayerName','XTeam','pos')]
   names(sims) <- c('PlayerName','XTeam','pos')

# Replacing all the columns with zero in the start
   sims[is.na(sims)] <- 0

# finding total number of players according to their position
  x.k <- subset(x, pos == "k")
  x.qb <- subset(x, pos == "qb")
  x.rb <- subset(x, pos == "rb")
  x.te <- subset(x, pos == "te")
  x.wr <- subset(x, pos == "wr")
  n.k <- nrow(x.k)
  n.qb <- nrow(x.qb)
  n.rb <- nrow(x.rb)
  n.te <- nrow(x.te)
  n.wr <- nrow(x.wr)

# extracting residual matrix
  res.k <- residual.list[["residuals.k"]]
  res.qb <- residual.list[["residuals.qb"]]
  res.rb <- residual.list[["residuals.rb"]]
  res.te <- residual.list[["residuals.te"]]
  res.wr <- residual.list[["residuals.wr"]]

# Loop for nsims number of simulation
for (i in seq(1:nsims))
{
  sim <- data.frame(matrix(ncol = 3, nrow = 0))
  sim[1:nrow(x),1:3] <- x[1:nrow(x),c('PlayerName','XTeam','pos')]
  names(sim) <- c('PlayerName','XTeam','pos')
# sampling the matrix for getting nrow(x1) amount of data from 1 to 20 quarter backs residual
  sample.k <- res.k[sample(nrow(res.k), size = n.k, replace = TRUE),]
# sample.k[sample.k < 0 ] = 0
  sample.qb <- res.qb[sample(nrow(res.qb), size = n.qb, replace = TRUE),]
# sample.qb[sample.qb < 0 ] = 0
  sample.rb <- res.rb[sample(nrow(res.rb), size = n.rb, replace = TRUE),]
# sample.rb[sample.rb < 0 ] = 0
  sample.te <- res.te[sample(nrow(res.te), size = n.te, replace = TRUE),]
# sample.te[sample.te < 0 ] = 0
  sample.wr <- res.wr[sample(nrow(res.wr), size = n.wr, replace = TRUE),]
# sample.wr[sample.wr < 0 ] = 0

# Adding it back to the stats
```

```
# combining residual data frame
  stats.reg <- rbind(sample.k, sample.qb, sample.rb, sample.te, sample.wr)
#  stats.reg <- cbind(PlayerName = stats$PlayerName, XTeam = stats$XTeam, pos = stats$pos, stats.reg)
# getting only the values for projected data frame
  stats1 <- subset(stats, select = c(names(stats.reg)))

# adding them up and changing negative values to 0
  stats.add <- stats1 + stats.reg
  stats.add[stats.add < 0] = 0

# final stats with modified parameter values, playername, team and position
  stats.final <- cbind(PlayerName = stats$PlayerName, XTeam = stats$XTeam, pos = stats$pos, stats.add)

# getting the value for nteam, cap, pos and points
  nteams <- league.object[["nteams"]]
  cap <- league.object[["cap"]]
  pos <- league.object[["pos"]]
  points <- league.object[["points"]]

# getting the league object to get the value column
  league.object <- league(stats.final, nteams, cap, pos, points)

# extracting the value column values and storing it in sims to corresponding names
  values <- league.object[["values"]]
  subset.values <- subset(values, select = c('PlayerName','value'))

# used sim as dummy to merge and then reassigned it
  sim <- merge (sim, subset.values, by = 'PlayerName', all.x = T)
  sims[1:nrow(sims),(i+3)] = sim[1:nrow(sims) ,4]

# Replacing all the columns with zero in the start
   sims[is.na(sims)] <- 0
  }
  return(sims)
}
```

```
# Adding it to current league object
nsims = 100
rng.seed = 1000
league.object[["sims"]] <- addNoise(league.object, residual.list, nsims, rng.seed)
```

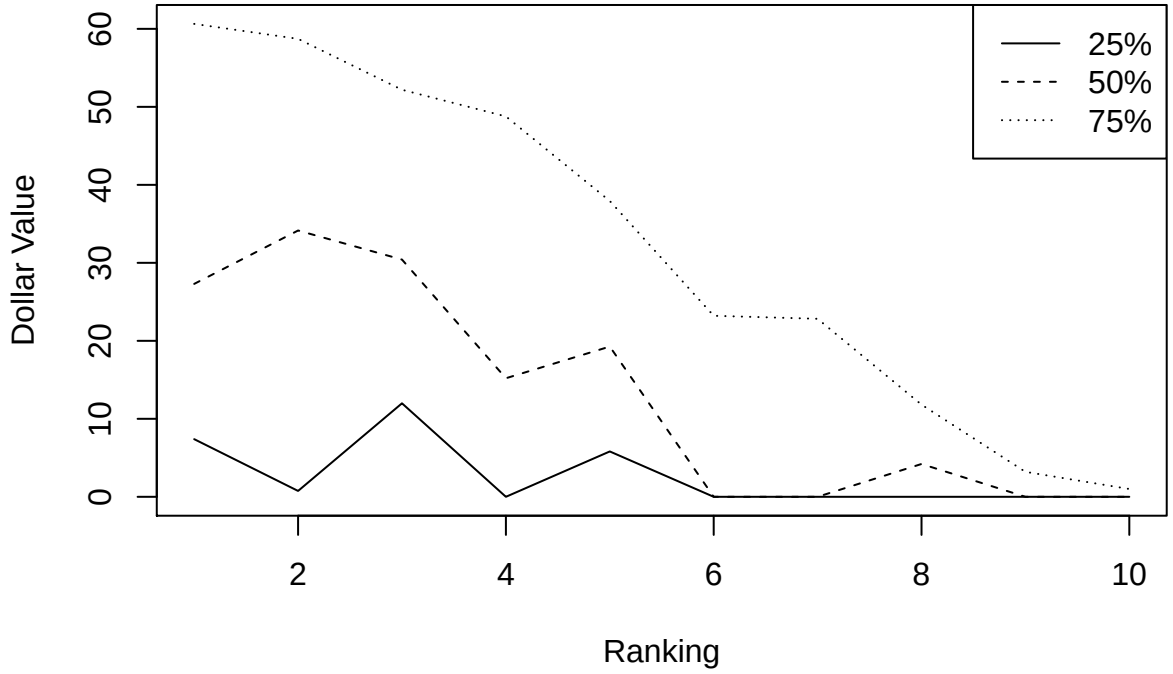
```
# Developing the quantile function for class league

quantile.league <- function(league.object, probs, class = "league")
{
  sims <- league.object[["sims"]]
  quant <- data.frame(t(apply(sims[,4:ncol(sims)],1,quantile, probs)))
  quant <- cbind(PlayerName = sims$PlayerName, XTeam = sims$XTeam,
                 pos = sims$pos, quant)
}

probs <- c(0.25, 0.5, 0.75)
quant <- quantile(league.object, probs)
```

```r
# Developing the confidence interval of class type league.conf.interval

conf.interval <- function (league.object, probs)
{
# finding number of teams and position
  nteams <- league.object[["nteams"]]
  pos <- league.object[["pos"]]

# calling the function quantile
  quant <- quantile(league.object, probs)

# subsetting the data on the basis of position
  quant.k <- subset(quant, pos == "k")
  quant.qb <- subset(quant, pos == "qb")
  quant.rb <- subset(quant, pos == "rb")
  quant.te <- subset(quant, pos == "te")
  quant.wr <- subset(quant, pos == "wr")

# ordering the data on the basis of last column
  l.col <- ncol(quant)
  quant.k <- quant.k[ order(quant.k[,l.col], decreasing = TRUE), ]
  quant.qb <- quant.qb[ order(quant.qb[,l.col], decreasing = TRUE), ]
  quant.rb <- quant.rb[ order(quant.rb[,l.col], decreasing = TRUE), ]
  quant.te <- quant.te[ order(quant.te[,l.col], decreasing = TRUE), ]
  quant.wr <- quant.wr[ order(quant.wr[,l.col], decreasing = TRUE), ]

# restricting the rows to number of required players
  quant.k <- quant.k[(seq(nteams*pos[["k"]])), ]
  quant.qb <- quant.qb[(seq(nteams*pos[["qb"]])), ]
  quant.rb <- quant.rb[(seq(nteams*pos[["rb"]])), ]
  quant.te <- quant.te[(seq(nteams*pos[["te"]])), ]
  quant.wr <- quant.wr[(seq(nteams*pos[["wr"]])), ]

list.ci <- list(k = quant.k, qb = quant.qb, rb = quant.rb,
                te = quant.te, wr = quant.wr)
class(list.ci) <- "league.conf.interval"
return(list.ci)
}
```

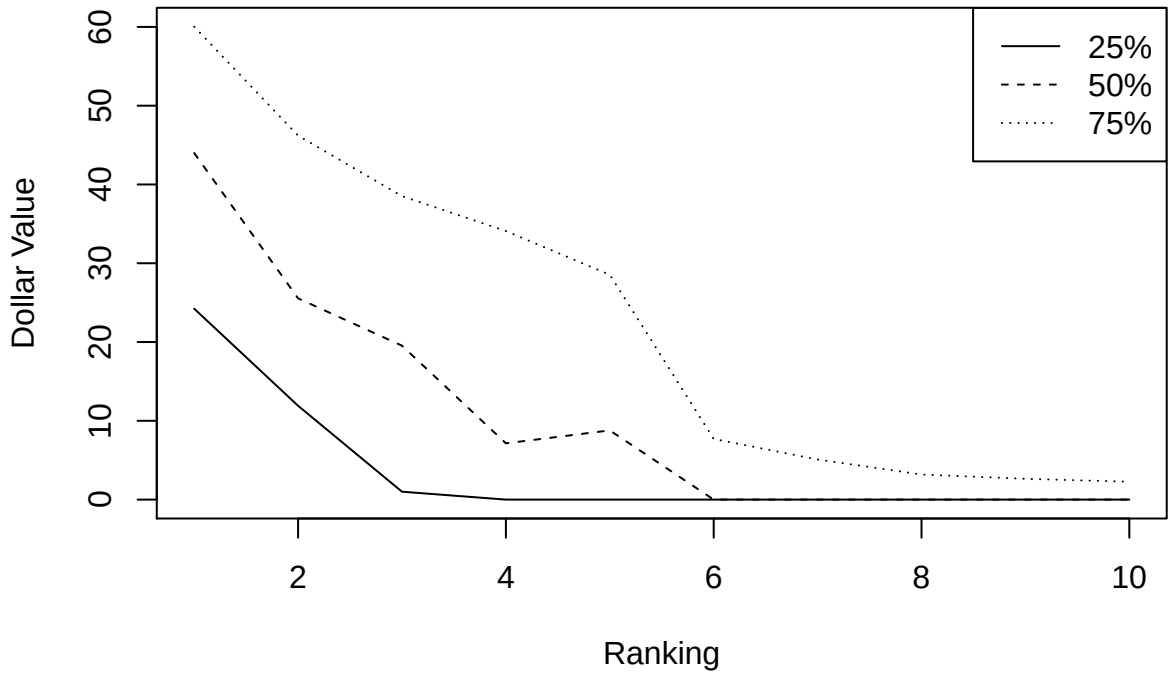
```r
plot.league.conf.interval <- function(l, pos, class = "league.conf.interval")
{
# reading list for corresponding position
  l1 <- l[[pos]]
  len <- nrow(l1)

# stating x and y for 25% 50% and 75% probability
  x <- seq(len)
  y1 <- l1[,4]
  y2 <- l1[,5]
  y3 <- l1[,6]

# finding ymax for y scale, plotting and assigning legend
  ymax <- max(y1[1], y2[1], y3[1])
```

```
  plot(x, y1, ylim = range(c(ymax,0)), xlab = "Ranking",
       ylab = "Dollar Value", type="l",lty = 1)
  lines(x, y2, ylim = range(c(ymax,0)), type="l",lty = 2)
  lines(x, y3, ylim = range(c(ymax,0)), type="l",lty = 3)
  legend('topright', c('25%','50%','75%'), lty=c(1:3))
}
```

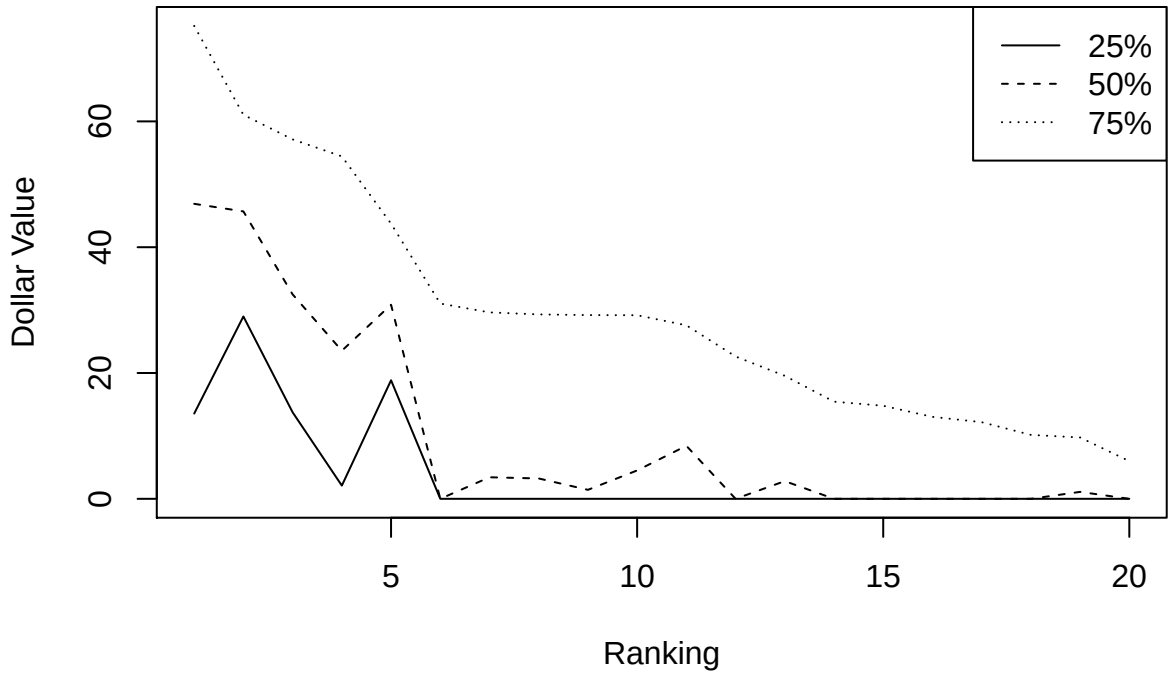
```
# Calling the plot function for class league.conf.interval and plot

# defining probs
probs <- c(0.25, 0.5, 0.75)

# calling conf.interval
l <- conf.interval(league.object, probs)

# for kickers
plot(l, 'k')
```



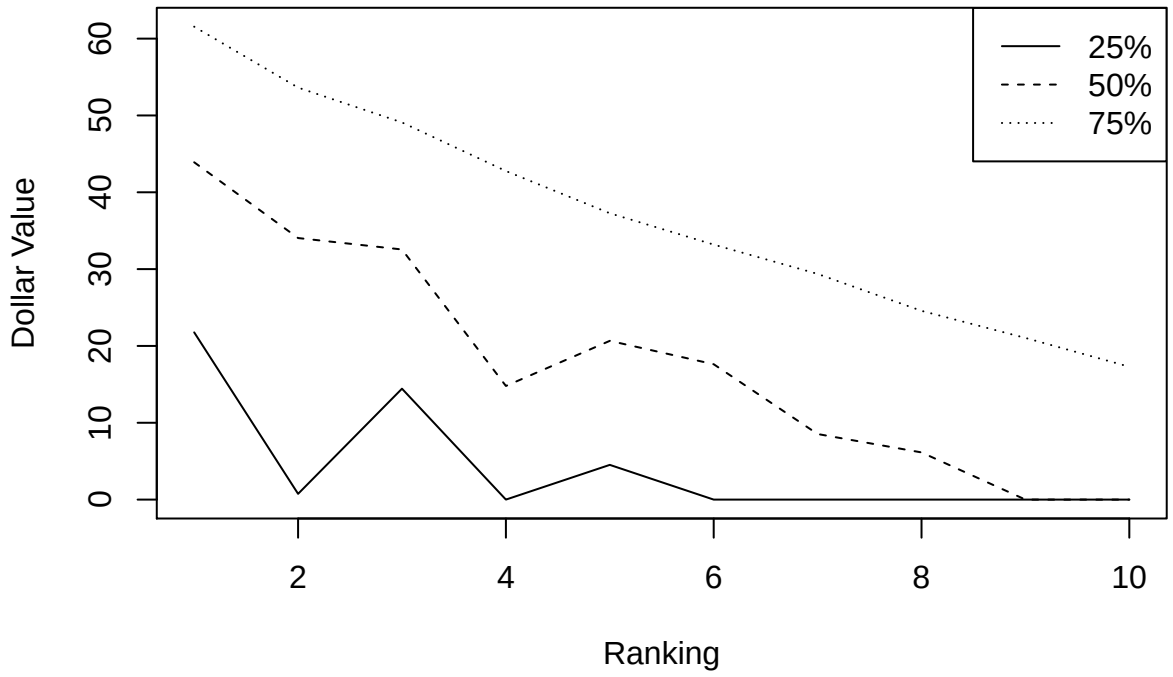

```
# for qb
plot(l, 'qb')
```




```
# for rb
plot(l, 'rb')
```




```
# for te
plot(l, 'te')
```




```
# for wr
plot(l, 'wr')
```