Stat Comp Final

Michael Ackerman

November 23, 2015

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.

1. 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.

```
path = "~/Documents/football-values-master-2/2015/"
file='outfile.csv'
  k <- read.csv(file.path(path, 'proj_k15.csv'), header=TRUE, stringsAsFactors=FALSE)
  qb <- read.csv(file.path(path, 'proj_qb15.csv'), header=TRUE, stringsAsFactors=FALSE)
  rb <- read.csv(file.path(path, 'proj_rb15.csv'), header=TRUE, stringsAsFactors=FALSE)
  te <- read.csv(file.path(path, 'proj_te15.csv'), header=TRUE, stringsAsFactors=FALSE)
  wr <- read.csv(file.path(path, 'proj_wr15.csv'), header=TRUE, stringsAsFactors=FALSE)
observed <- read.csv(file.path(path, 'nfl_current15.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
  x <- rbind(k[,cols], qb[,cols], rb[,cols], te[,cols], wr[,cols])
  # furthermore, gsub does global replacement, not single replacement
  names(x) <- gsub('[.]', '', names(x))

  # bye-ajusted values
  x[,c(-1:-2,-19)] <- (9/16)*x[,c(-1:-2,-19)]
  nobye <- which(x[,'Team'] == c('CLE', 'NO', 'NYG', 'PIT'))
  x[nobye, c(-1:-2,-19)]  <- (10/9)*x[nobye,c(-1:-2,-19)]

x <- x[order(x$fpts, decreasing=T),]
posis <- unique(x$pos)
numb <- c(20,40,60,20,20) #amount of each position we want

y <- NULL # Remove the extra players we dont care to look at
for(i in seq_along(posis)){
y <- rbind(y, x[which(x$pos==posis[i])[1:numb[i]],])
}

# Merge Observed and predicted data
x <- y[order(y$fpts, decreasing=T),]
pro <- x
df <- merge(x, observed, by.x = "PlayerName", by.y = "Name", all.x = T, all.y =F)
df <- df[order(df$fpts, decreasing=T),]
rownames(df) <- NULL
df[is.na(df)] <- 0

name <- c('field goals','field goals attempted','extra points','passing attempts','passing completions'

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")

diff <- cbind( df[,'pos'], df[,observed_col]-df[,projected_col] )
names(diff) <- c('pos', name)

ldiff <- split(diff[,2:16], diff[,'pos']) #list of matrices by 'pos'
```

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.

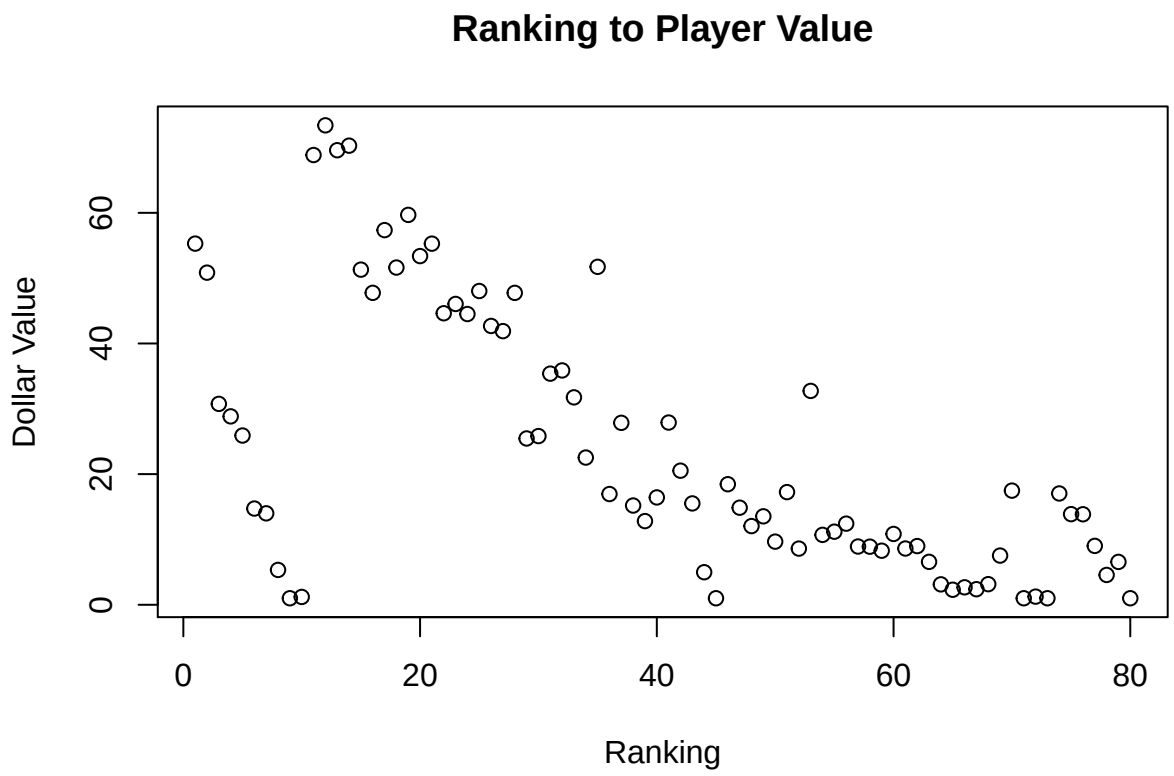
4. Create a `print` method for the league class. It should print the players and dollar values (you may choose to only include players with values greater than $0).

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

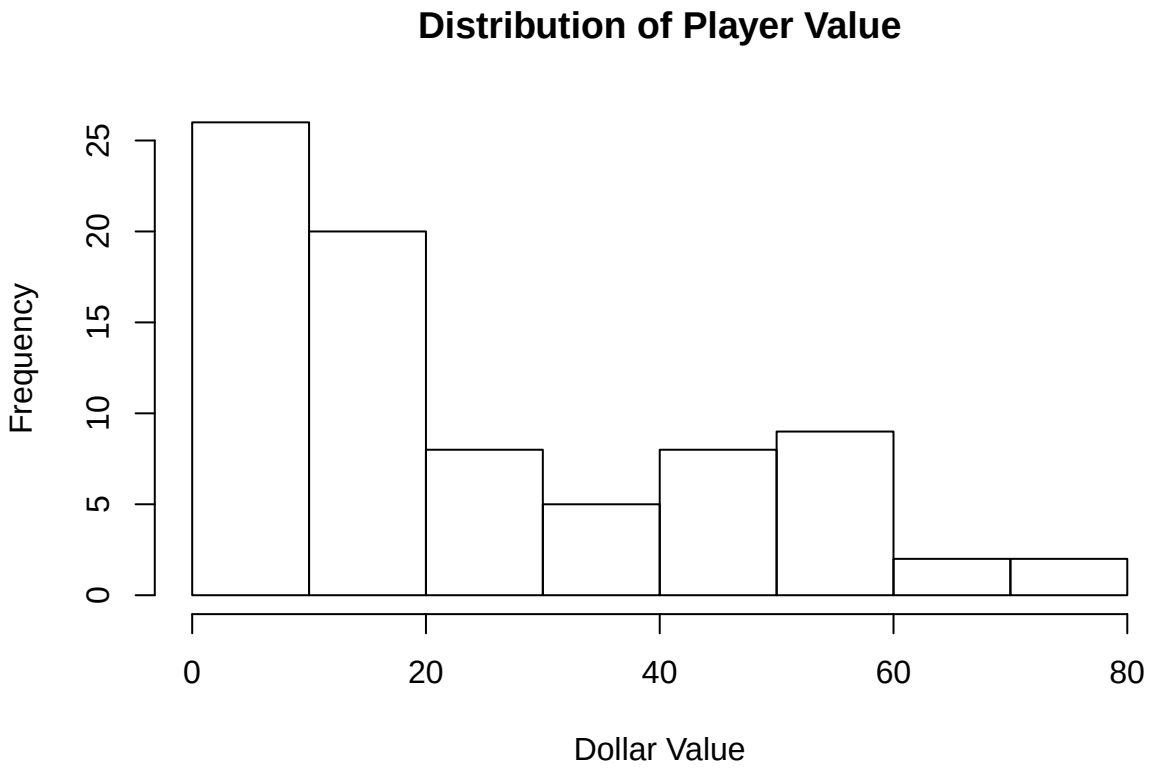
```
#league function will create a list where the first element are the stats and proceeding elements are t

#the following functions will pull the elements of that list and computed the desired values, adding th

league <- function(stats, nTeams=12, cap=200,posReq=c(qb=1, rb=2, wr=3, te=1, k=1),
                     points=c(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
                     rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)){
 if(is.list(posReq)) posReq <- unlist(posReq)
 if(is.list(points)) points <- unlist(points)
  info <- list(stats=stats, nTeams=nTeams, cap=cap, posReq=posReq, points=points)
  class(info) <- 'league'
  return(info)
}
###################################################################

calcPoints<- function(p){UseMethod("calcPoints")}
buildValues <- function(p){UseMethod("buildValues")}

calcPoints.league <- function(p){ #calculate point
  x <- y <- p[[1]]

  for(i in names(p$points)) {
    x[,sprintf("p_%s", i)] <- x[,i]*p$points[[i]]    }

  y[,'points'] <- rowSums(x[,grep("^p_", names(x))])
  p[[1]] <- y
  return(p)
}
###################################################################

buildValues.league <- function(stats){ #calculate dollar values
  p<- calcPoints(stats)
  x<- p[[1]]
  x2 <- x[order(x[,'points'], decreasing=TRUE),]
  # calculate marginal points by subtracting "baseline" player's points
  for(i in names(p$posReq)) {
    ix <- which(x2[,'pos'] == i)
    baseline <- p$posReq[[i]]*p$nTeams
    if(baseline == 0) {
      x2[ix, 'marg'] <- -1
    } else {
      x2[ix, 'marg'] <- x2[ix,'points'] - x2[ix[baseline],'points']
    }
  }
  x3 <- x2
  x3[,'value'] <-  ifelse( x2[,'marg'] >= 0,
   x3[,'marg']*(p$nTeams*p$cap-nrow(x3))/sum(x3[which(x3[,'marg']>0),'marg']) +1, 0)
```

```
  x3[is.na(x3[,'value']),] <- 0

  x3[,'marg'] <- NULL
  x4 <- x3[rownames(x), ]
  p[[1]]<-x4
  return(p)
}
#p <-buildValues(l)
################################################
print.league <- function(stats){
  x <-buildValues(stats)$stats
  x <- x[x[,'value']>0,]
  cat(paste("The",x$pos,  x$PlayerName, "is worth: $", round(x$value,2), "\n"))
}
################################################
plot.league <- function(p){
  x <- buildValues(p)$stats
    x <- x[x[,'value']>0,]
  plot(x$value, ylab="Dollar Value", xlab="Ranking", main="Ranking to Player Value")
}
################################################
hist.league <- function(p){
  x <- buildValues(p)$stats
  x <- x[x[,'value']>0,]
  hist(x$value, xlab= "Dollar Value", main= "Distribution of Player Value")
}
################################################
boxplot.league <- function(p){
  x <- buildValues(p)$stats
  x <- x[x[,'value']>0,c('pos','value')]
  boxplot(value~pos, data= x)
}
#rm(print.league)
```

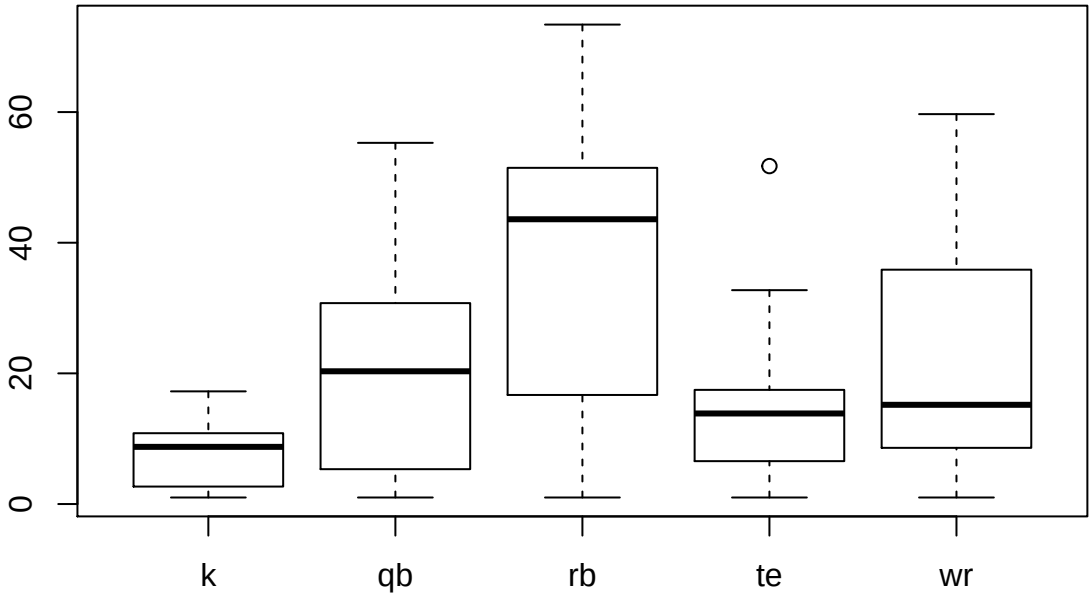
```
pos <- list(qb=1, rb=2, wr=3, te=1, k=1)
pnts <- list(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
             rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)
l <- league(stats=df, nTeams=10, cap=200, posReq=pos, points=pnts)

plot(l)
```


Ranking to Player Value
Dollar Value
0
20
40
60
0
20
40
60
80
Ranking

```
hist(l)
```


Distribution of Player Value
Frequency
0
5
10
15
20
25
0
20
40
60
80
Dollar Value

```
boxplot(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.

```
simValue1 <- function(...){UseMethod("simValue1")}

simValue1.league <- function(stats,resid){
  x<- stats$stats
  y <- x[,c('PlayerName', 'fpts')] #use later to reorder
  lx <- split(x, x[,'pos']) # yes thats a "L"

  lnew <- list()
  new1 <- NULL
  nums <- sapply(x, is.numeric)

  for(i in 1:5){
  posis <- names(lx[i])

 rowi <- floor(runif(dim(lx[[i]])[1], min = 1, max= dim(resid[[posis]])[1] ))

 x2 <- lx[[i]][,c("fg","fga","xpt","pass_att","pass_cmp","pass_yds","pass_tds","pass_ints","rush_att","r

  lnew[[i]] <- cbind(lx[[i]][,c('PlayerName','pos')], x2+resid[[posis]][rowi,])
  colnames(lnew[[i]]) <- c('PlayerName','pos', colnames(x2))

  new1 <- rbind(new1 ,lnew[[i]])
  }

 new1[new1<0] <- 0
new1 <-  merge(y,new1, by = "PlayerName", all.x= FALSE,all.y=T, sort=FALSE)#restore original order
stats$stats <- new1

 val <-buildValues(stats)[[1]]$value
return(val)
  }
#########################################################
#simValue1(l, ldiff)

addNoise <- function(p,...){UseMethod("addNoise")}
addNoise.league <- function(stats, resid= ldiff, nsims=100, seed=120, warned = FALSE){
  if(nsims>4000 & warned == 0){warning("This is going to take a little while. Are you sure you want to c
  set.seed(seed)
  simulations <- matrix(NA, nrow=length(simValue1(stats, resid)) , ncol= nsims)
  for(i in 1:nsims){
  simulations[,i] <- simValue1(stats, resid)
  }
stats$simulations <- simulations
return(stats)
}
#########################################################
#j <- addNoise(stats, ldiff, 100,,T)

quantile.league <- function(stats, probs= c(0.25, 0.5, 0.75),resid= ldiff, nsims=100){
```

```
  if(is.null(stats$simulations)){
    stats = addNoise(stats,resid, nsims)
    message(paste("This is a simulation calculated by this function becuase you didn't provide one yours
  }
  quant1 <- apply(stats$simulations, 1, quantile, probs)
  quant <- quant1[,which(quant1[2,]>0)] #Only the good players
 return(quant1)
}
#########################################################
#quantile(l,probs= c(0.25, 0.5, 0.75))

conf.interval <- function(...){UseMethod("conf.interval")}

conf.interval.league <- function(stats, probs= c(0.25, 0.5, 0.75),resid= ldiff, nsims=100){
  #quantile(stats,probs= c(0.25, 0.5, 0.75),resid= ldiff, nsims=100)
  x <- stats$stats
  q <- quantile(stats)
  q <- t(q)

  H<- data.frame(PlayerName=x$PlayerName, pos=x$pos, lower=q[,1], mid=q[,2], upper=q[,3])
  H <- H[order(H[,'upper'], decreasing=TRUE),]
H <- split(H, H$pos)
n <- stats$nTeams*stats$posReq
for(i in names(n)){
  H[[i]] <- H[[i]][1:n[[i]],]
  rownames(H[[i]])<- NULL
}
Conf.int <- H
  class(Conf.int) <- 'league.conf.interval'
return(Conf.int)
}
#########################################################
#conf.interval(l)

plot.league.conf.interval <- function(intervals, pos="qb"){
  d <- intervals[[pos]]
  plot(d$m, ylab= "Value", xlab= "Per Player")
  lines(d$l)
  lines(d$u)
}
```

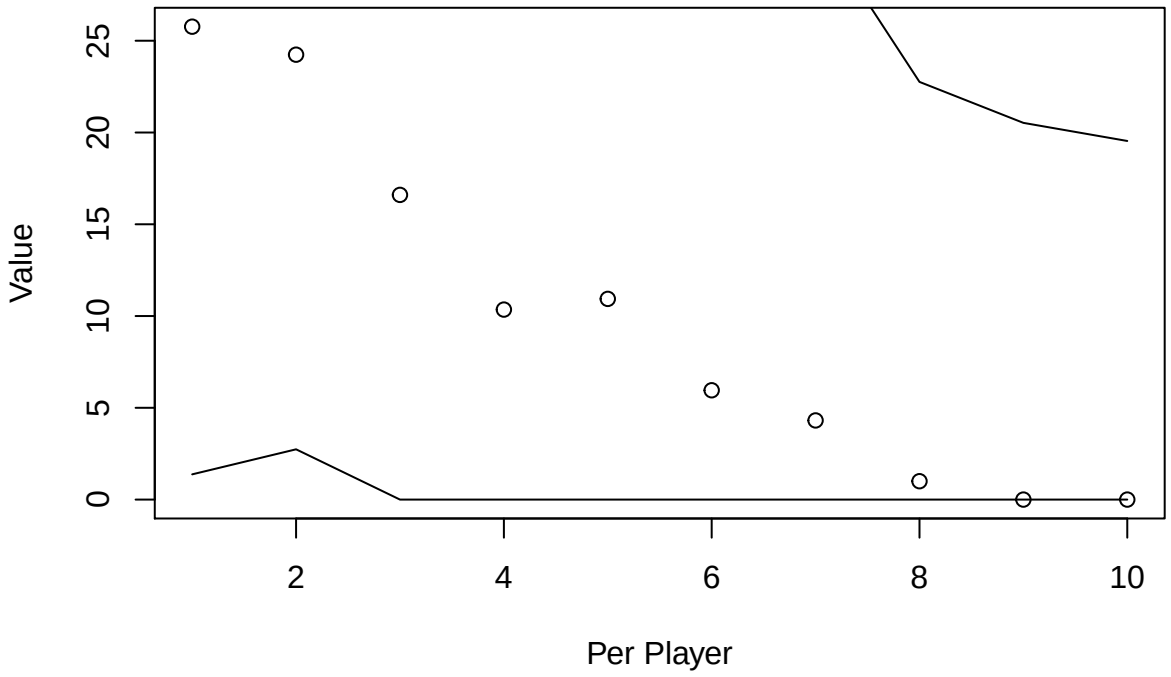
```
l1 <- addNoise(l, ldiff, 1000)
#quantile(l1)
(ci <- conf.interval(l1) )
```

```
## $k
##            PlayerName pos lower      mid    upper
## 1  Stephen Gostkowski   k     0 6.878139 25.41617
## 2       Justin Tucker   k     0 4.545344 22.93182
## 3     Steven Hauschka   k     0 2.757110 21.08041
## 4        Mason Crosby   k     0 2.469716 20.75701
## 5        Connor Barth   k     0 1.186462 20.21942
## 6         Cody Parkey   k     0 1.237034 19.87219
```

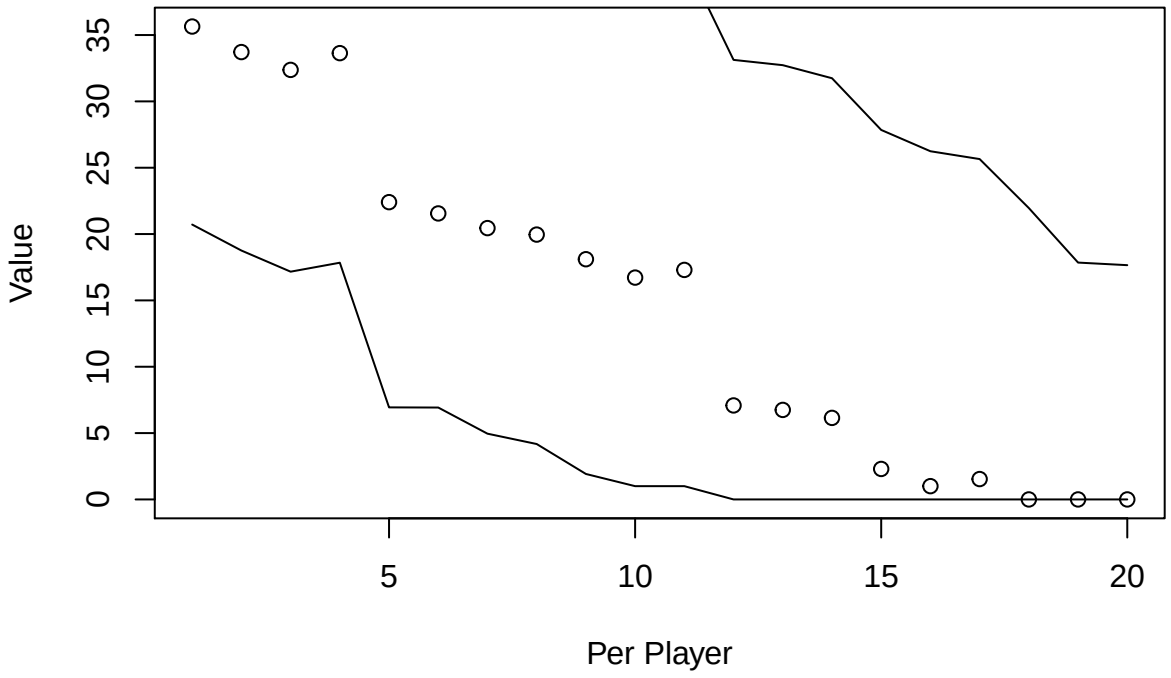
```
## 7      Adam Vinatieri   k     0 1.000000 17.46042
## 8          Dan Bailey   k     0 1.000000 16.07457
## 9     Garrett Hartley   k     0 0.000000 15.22240
## 10        Matt Bryant   k     0 1.000000 15.19167
##
## $qb
##            PlayerName pos    lower       mid    upper
## 1         Andrew Luck  qb 1.374142 25.765582 52.94515
## 2       Aaron Rodgers  qb 2.738317 24.240042 49.31976
## 3      Peyton Manning  qb 0.000000 16.600434 42.81810
## 4          Drew Brees  qb 0.000000 10.352261 37.64642
## 5      Russell Wilson  qb 0.000000 10.934662 37.45977
## 6         Eli Manning  qb 0.000000  5.954975 31.56775
## 7           Matt Ryan  qb 0.000000  4.312203 31.41462
## 8  Ben Roethlisberger  qb 0.000000  1.000000 22.75736
## 9          Cam Newton  qb 0.000000  0.000000 20.52485
## 10          Tony Romo  qb 0.000000  0.000000 19.53810
##
## $rb
##           PlayerName pos     lower       mid    upper
## 1     Marshawn Lynch  rb 20.716386 35.640261 60.64100
## 2    Adrian Peterson  rb 18.757873 33.715747 59.04895
## 3     Jamaal Charles  rb 17.165714 32.372994 57.87441
## 4         Eddie Lacy  rb 17.840058 33.634734 57.49536
## 5       Le'Veon Bell  rb  6.936886 22.405458 47.79583
## 6      C.J. Anderson  rb  6.924751 21.554476 45.20080
## 7         Matt Forte  rb  4.955950 20.448858 45.08398
## 8       LeSean McCoy  rb  4.175209 19.961172 45.00572
## 9     DeMarco Murray  rb  1.917984 18.103001 42.65922
## 10       Mark Ingram  rb  1.000000 16.719314 41.75183
## 11       Jeremy Hill  rb  1.000000 17.296928 40.76846
## 12      Lamar Miller  rb  0.000000  7.083389 33.12705
## 13    Justin Forsett  rb  0.000000  6.746294 32.72936
## 14     Alfred Morris  rb  0.000000  6.145185 31.74454
## 15       Carlos Hyde  rb  0.000000  2.295532 27.84763
## 16        Frank Gore  rb  0.000000  1.000000 26.24185
## 17     Melvin Gordon  rb  0.000000  1.534641 25.65163
## 18   Latavius Murray  rb  0.000000  0.000000 21.95739
## 19 LeGarrette Blount  rb  0.000000  0.000000 17.85114
## 20     Joseph Randle  rb  0.000000  0.000000 17.65479
##
## $te
##           PlayerName pos     lower       mid     upper
## 1     Rob Gronkowski  te 19.162729 26.535133 41.261380
## 2       Jimmy Graham  te  8.829552 16.705829 31.230516
## 3       Jason Witten  te  0.000000  4.616731 20.976896
## 4       Travis Kelce  te  0.000000  6.737512 20.504769
## 5         Greg Olsen  te  0.000000  6.128761 20.259057
## 6       Dwayne Allen  te  0.000000  1.000000 14.136941
## 7  Martellus Bennett  te  0.000000  3.252888 13.582451
## 8      Julius Thomas  te  0.000000  1.000000 11.559316
## 9          Zach Ertz  te  0.000000  0.000000 10.051224
## 10     Antonio Gates  te  0.000000  0.000000  9.487268
##
```

```
## $wr
##           PlayerName pos     lower       mid    upper
## 1   Demaryius Thomas  wr 17.423942 31.731078 49.08055
## 2      Antonio Brown  wr 14.998097 29.779138 45.44572
## 3         Dez Bryant  wr 12.818175 27.742894 44.71384
## 4  Odell Beckham Jr.  wr 14.422451 28.631424 44.05316
## 5     Calvin Johnson  wr  9.812723 24.151895 42.11393
## 6       Randall Cobb  wr 10.197327 23.135804 39.65153
## 7        Julio Jones  wr  5.225330 19.796613 37.05584
## 8         Mike Evans  wr  0.000000 14.602990 31.83806
## 9     Alshon Jeffery  wr  1.594490 15.627932 31.32261
## 10        A.J. Green  wr  1.830076 14.619778 31.18591
## 11       T.Y. Hilton  wr  0.000000 12.949688 29.92034
## 12  Emmanuel Sanders  wr  0.000000  9.812911 27.31990
## 13     Brandin Cooks  wr  0.000000  7.494311 25.79536
## 14   Jordan Matthews  wr  0.000000  5.666667 22.06691
## 15   DeAndre Hopkins  wr  0.000000  4.000544 20.51092
## 16    Julian Edelman  wr  0.000000  2.338145 19.94925
## 17     Andre Johnson  wr  0.000000  2.519564 19.56885
## 18     Jeremy Maclin  wr  0.000000  0.000000 19.52183
## 19    DeSean Jackson  wr  0.000000  1.055384 18.09290
## 20  Brandon Marshall  wr  0.000000  0.000000 17.67164
## 21       Golden Tate  wr  0.000000  0.000000 17.54724
## 22     Davante Adams  wr  0.000000  3.165195 17.44285
## 23     Sammy Watkins  wr  0.000000  0.000000 17.05039
## 24      Keenan Allen  wr  0.000000  0.000000 16.99805
## 25      Mike Wallace  wr  0.000000  0.000000 16.57883
## 26     Michael Floyd  wr  0.000000  0.000000 14.15705
## 27   Martavis Bryant  wr  0.000000  0.000000 14.10349
## 28   Vincent Jackson  wr  0.000000  0.000000 13.35935
## 29    Allen Robinson  wr  0.000000  0.000000 13.09572
## 30   Charles Johnson  wr  0.000000  0.000000 12.47666
##
## attr(,"class")
## [1] "league.conf.interval"
```

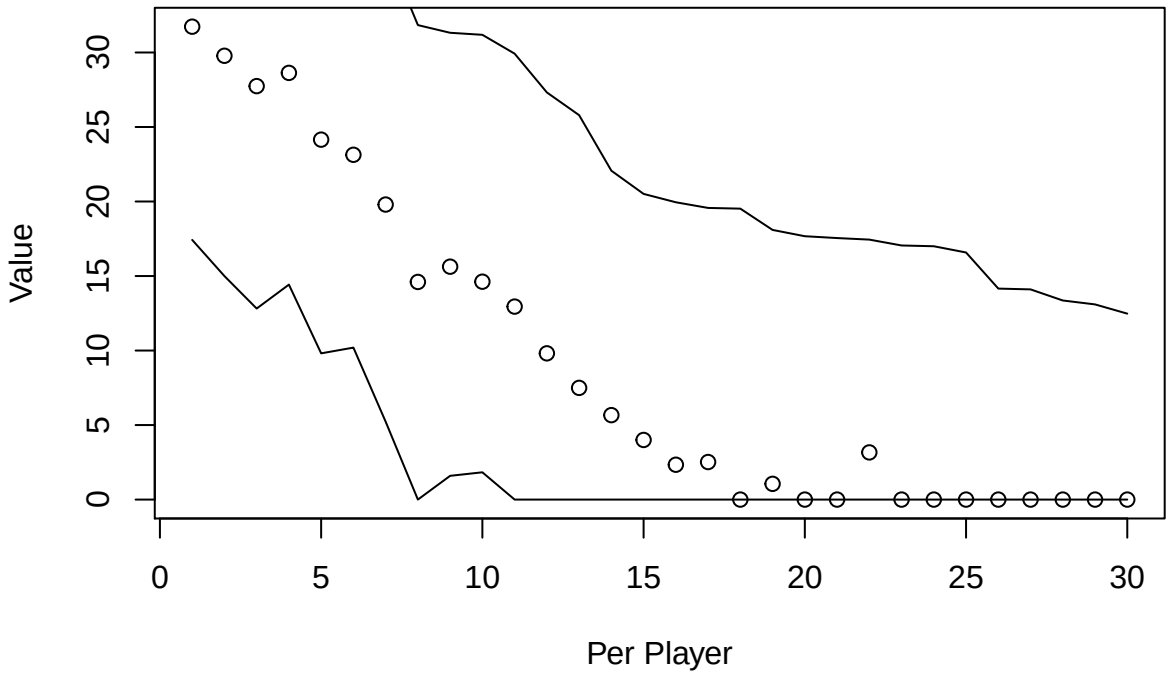
```
plot(ci, 'qb')
```



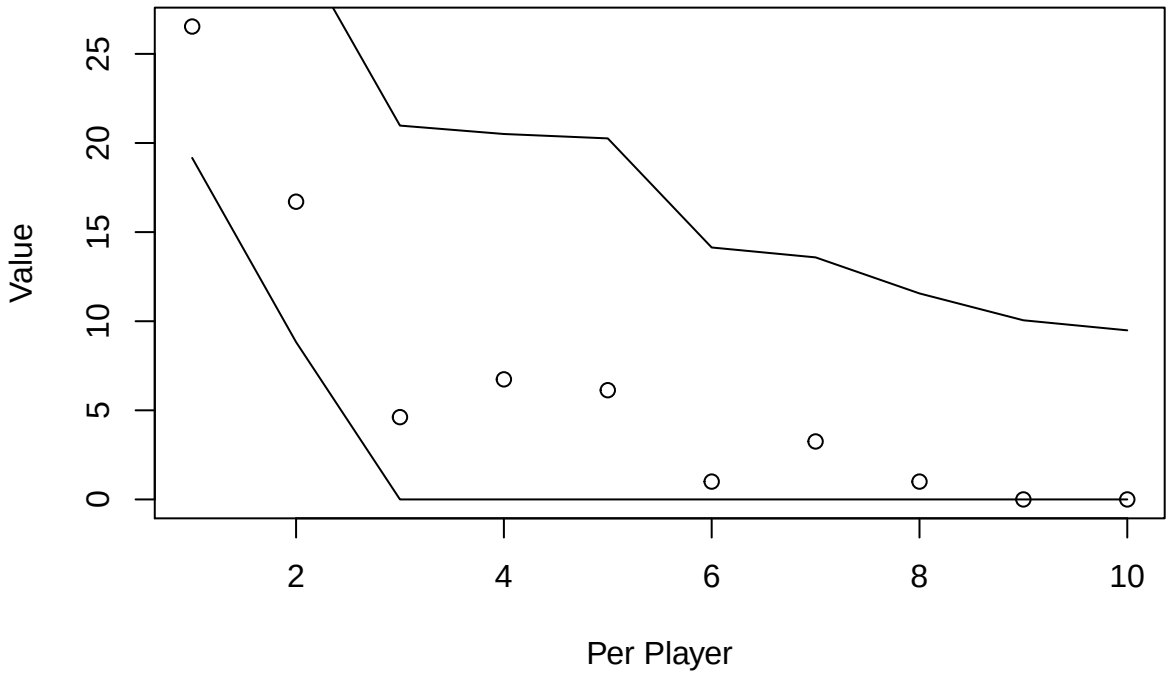

```
plot(ci, 'rb')
```



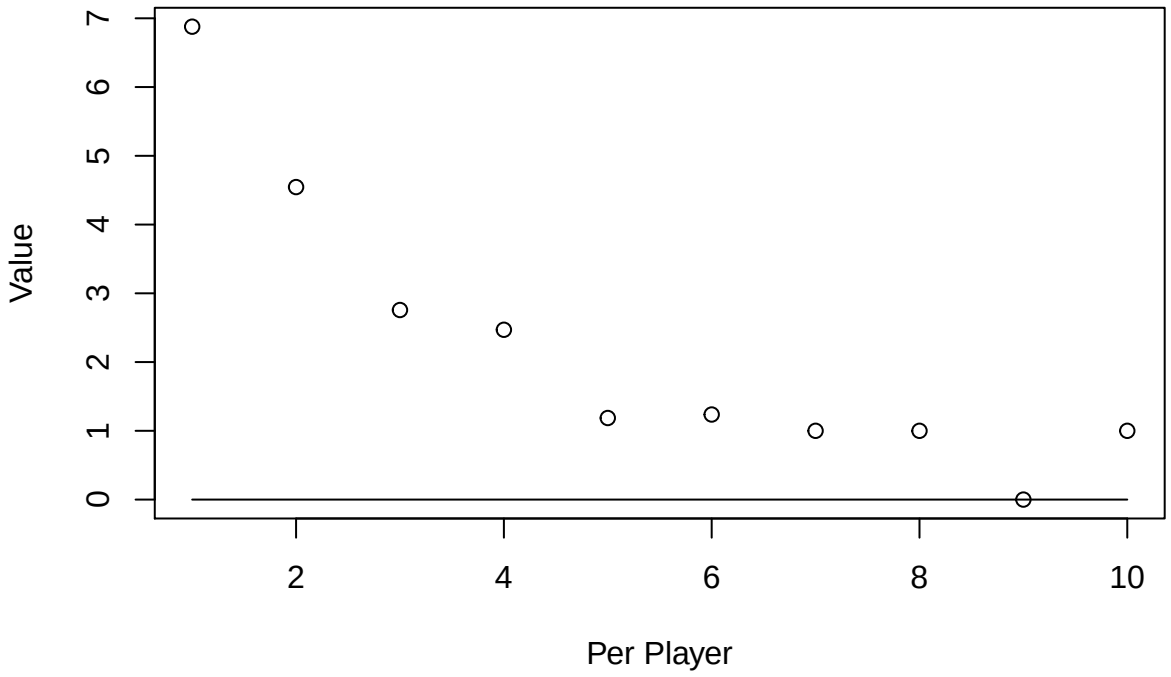

```
plot(ci, 'wr')
```




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




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


Value
0
1
2
3
4
5
6
7
2
4
6
8
10
Per Player