Bios 6301: Final Project

Brooklyn Stanley

12/14/2015

Due Monday, 14 December, 6:00 PM

200 points total.

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

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

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

Task 1: Finding Residuals (80 points)

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

1. Read and combine the projection data (five files) into one data set, adding a position column.
2. The NFL season is 17 weeks long, and 10 weeks have been completed. Each team plays 16 games and has one week off, called the bye week. Four teams have yet to have their bye week: CLE, NO, NYG, PIT. These four teams have played ten games, and every other team has played nine games. Multiply the numeric columns in the projection data by the percentage of games played (for example, 10/16 if team is PIT).
3. Sort and order the data by the `fpts` column descendingly. Subset the data by keeping the top 20 kickers, top 20 quarterbacks, top 40 running backs, top 60 wide recievers, and top 20 tight ends. Thus the projection data should only have 160 rows.
4. Read in the observed data (`nfl_current15.csv`)
5. Merge the projected data with the observed data by the player's name. Keep all 160 rows from the projection data. If observed data is missing, set it to zero.
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.

```
setwd("~/Documents/Biostatistics/Bios 301/Bios6301-Homework")

## Question 1

k <- read.csv(file.path(getwd(), '2015/proj_k15.csv'), header=TRUE, stringsAsFactors=FALSE)
qb <- read.csv(file.path(getwd(), '2015/proj_qb15.csv'), header=TRUE, stringsAsFactors=FALSE)
rb <- read.csv(file.path(getwd(), '2015/proj_rb15.csv'), header=TRUE, stringsAsFactors=FALSE)
te <- read.csv(file.path(getwd(), '2015/proj_te15.csv'), header=TRUE, stringsAsFactors=FALSE)
wr <- read.csv(file.path(getwd(), '2015/proj_wr15.csv'), header=TRUE, stringsAsFactors=FALSE)
```

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

## Question 2

indicesTen <- (projData[,2]=='CLE' | projData[,2]=='NO' | projData[,2]=='NYG' | projData[,2]=='PIT')
projData[indicesTen, 3:18] <- projData[indicesTen, 3:18] * 10/16
projData[!indicesTen, 3:18] <- projData[!indicesTen, 3:18] * 9/16

## Question 3

# Remove 3 players with missing observed data
projData <- projData[-which(projData$PlayerName == "Garrett Hartley"),]
projData <- projData[-which(projData$PlayerName == "Breshad Perriman"),]
projData <- projData[-which(projData$PlayerName == "Victor Cruz"),]

projData <- projData[order(-projData$fpts), ]
projData_top <- projData[1,]
qb_C <- 1
k_C <- 0
rb_C <- 0
wr_C <- 0
te_C <- 0
for (i in 2:length(projData$PlayerName)){
  if (length(projData_top$PlayerName) == 160){
    break
  }
  if (projData$pos[i] == 'qb' & qb_C < 20){
    projData_top <- rbind(projData_top, projData[i,])
    qb_C <- qb_C + 1
  }
  if (projData$pos[i] == 'k' & k_C < 20){
    projData_top <- rbind(projData_top, projData[i,])
    k_C <- k_C + 1
  }
  if (projData$pos[i] == 'rb' & rb_C < 40){
```

```
    projData_top <- rbind(projData_top, projData[i,])
    rb_C <- rb_C + 1
  }
  if (projData$pos[i] == 'wr' & wr_C < 60){
    projData_top <- rbind(projData_top, projData[i,])
    wr_C <- wr_C + 1
  }
  if (projData$pos[i] == 'te' & te_C < 20){
    projData_top <- rbind(projData_top, projData[i,])
    te_C <- te_C + 1
  }
}
row.names(projData_top) <- NULL

## Question 4

current <- read.csv(file.path(getwd(), '2015/nfl_current15.csv'), header=TRUE,
                    stringsAsFactors=FALSE)

## Question 5

fullData <- merge(projData_top, current, by.x='PlayerName', by.y='Name', all.x=T, all.y=F)

## Question 6

fullData$FG <- fullData$FGM - fullData$fg
fullData$FG.Att <- fullData$FGA - fullData$fga
fullData$XPT <- fullData$xpt - fullData$XPM
fullData$P.Att <- fullData$Att.pass - fullData$pass_att
fullData$P.Cmp <- fullData$Cmp.pass - fullData$pass_cmp
fullData$P.Yds <- fullData$Yds.pass - fullData$pass_yds
fullData$P.Tds <- fullData$TD.pass - fullData$pass_tds
fullData$P.Ints <- fullData$Int.pass - fullData$pass_ints
fullData$Ru.Att <- fullData$Att.rush - fullData$rush_att
fullData$Ru.Yds <- fullData$Yds.rush - fullData$rush_yds
fullData$Ru.tds <- fullData$TD.rush - fullData$rush_tds
fullData$Fum <- fullData$Fmb - fullData$fumbles
fullData$Re.Att <- fullData$Rec.catch - fullData$rec_att
fullData$Re.Yds <- fullData$Yds.catch - fullData$rec_yds
fullData$Re.Tds <- fullData$TD.catch - fullData$rec_tds

kicker <- fullData[fullData$pos == 'k',c(37:51)]
row.names(kicker) <- NULL
quarterBack <- fullData[fullData$pos == 'qb',c(37:51)]
row.names(quarterBack) <- NULL
runningBack <- fullData[fullData$pos == 'rb',c(37:51)]
row.names(runningBack) <- NULL
wideReceiver <- fullData[fullData$pos == 'wr',c(37:51)]
row.names(wideReceiver) <- NULL
tightEnd <- fullData[fullData$pos == 'te',c(37:51)]
row.names(tightEnd) <- NULL

football <- list(qb =quarterBack, rb =runningBack, wr=wideReceiver, te=tightEnd, k=kicker)
```

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

```
## Question 1

league <- function(stats, nTeams, cap, posReq, points){
  league <- list(stats=stats, nTeams=nTeams, cap=cap, posReq=posReq, points=points)
  class(league) <- "league"
  league
}

## Question 2

calcPoints <- function(league){
  # calculate new columns
  # convert NFL stat to fantasy points
  for(i in names(league$points)) {
    league$stats[,sprintf("p_%s", i)] <- league$stats[,i]*unlist(league$points[i])
  }

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

  # replace data with data ordered by points descendingly
  league$stats <- league$stats[order(-league$stats[,'points']),]
  return(league)
}

## Question 3

buildvalues <- function(league){
  # assumes points values have been calculated
```

```
  # calculate marginal points by subtracting "baseline" player's points
  for(i in names(league$posReq)) {
    ix <- which(league$stats[,'pos'] == i)
    baseline <- unlist(league$posReq[i])*league$nTeams
    if (baseline > sum(league$stats$pos == i)){
      stop(cat("ERROR... too many players requested for position: ",i))
    }
    if (!is.element(i,l$stats$pos)) {
      stop(cat("ERROR... player of type requested was not found in data: ",i))
    }
    if(baseline == 0) {
      league$stats[ix, 'marg'] <- -1
    } else {
      league$stats[ix, 'marg'] <- league$stats[ix,'points'] - league$stats[ix[baseline],'points']
    }
  }

  # calculation for player value
  margs <- league$stats[,'marg']
  margs[margs < 0] <- NA
  values <- margs*(league$nTeams*league$cap-length(margs))/sum(margs[!is.na(margs)]) + 1
  values[is.na(values)] <- 0
  league$stats[,"values"] <- values
  league$stats <- league$stats[order(league$stats[,'values'], decreasing=TRUE),]
  return(league)
}

## Question 4

print.league <- function(league){
  for (i in 1:length(league$stats[,1])){
    if (league$stats[i,'values'] > 0){
      cat(league$stats[i,'PlayerName'], ": ", league$stats[i,"values"])
    }
  }
}

## Question 5

plot.league <- function(league){
  x <- sum(league$stats[,'values'] > 0)
  values <- sort(league$stats[league$stats[,'values']>0,'values'], decreasing=T)
  plot(1:x, values, xlab="Ranking", ylab="Dollar Value",xlim=c(0,x))
}

## Question 6

boxplot.league <- function(league){
  plotter <- league$stats[league$stats[,'values'] > 0,]
  boxplot(values ~ pos, data=plotter, xlab="Position", ylab="Dollar Value")
}

## Question 7
```

```
hist.league <- function(league){
  plotter <- league$stats[league$stats[,'values'] > 0,]
  hist(plotter$values, main="League Histogram", xlab="Dollar Value")
}
```

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=projData, nTeams=10, cap=200, posReq=pos, points=pnts)
l <- calcPoints(l)
l <- buildvalues(l)
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 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.

```
## Question 1

addNoise <- function(league, res, nsims, seed=1234) {
  set.seed(seed)
  sims <- data.frame(PlayerName=league$stats$PlayerName, Team=league$stats$Team)
  error <- function(row, errMatrix){
    idx <- sample(1:length(errMatrix[,1]), 1)
    return(row + errMatrix[idx,])
  }
  for (i in 1:nsims){
    temp <- league
    kick <- temp$stats$pos == 'k'
    r_k <- football[['k']]
    k_err <- data.frame(matrix(unlist(apply(temp$stats[kick,c(3:5,7:18)], 1, error, r_k)),
                               nrow=sum(kick), byrow=T))
    k_err[k_err < 0] <- 0
    temp$stats[kick,c(3:5,7:18)] <- k_err
    quar <- temp$stats$pos == 'qb'
    r_qb <- football[['qb']]
    qb_err <- data.frame(matrix(unlist(apply(temp$stats[quar,c(3:5,7:18)], 1, error, r_qb)),
                               nrow=sum(quar), byrow=T))
    qb_err[qb_err < 0] <- 0
    temp$stats[quar,c(3:5,7:18)] <- qb_err
    wide <- temp$stats$pos == 'wr'
    r_wr <- football[['wr']]
    wr_err <- data.frame(matrix(unlist(apply(temp$stats[wide,c(3:5,7:18)], 1, error, r_wr)),
                               nrow=sum(wide), byrow=T))
    wr_err[wr_err < 0] <- 0
    temp$stats[wide,c(3:5,7:18)] <- wr_err
    tigh <- temp$stats$pos == 'te'
    r_te <- football[['te']]
    te_err <- data.frame(matrix(unlist(apply(temp$stats[tigh,c(3:5,7:18)], 1, error, r_te)),
                               nrow=sum(tigh), byrow=T))
    te_err[te_err < 0] <- 0
    temp$stats[tigh,c(3:5,7:18)] <- te_err
    runn <- temp$stats$pos == 'rb'
```

```
    r_rb <- football[['rb']]
    rb_err <- data.frame(matrix(unlist(apply(temp$stats[runn,c(3:5,7:18)], 1, error, r_rb)),
                                nrow=sum(runn), byrow=T))
    rb_err[rb_err < 0] <- 0
    temp$stats[runn,c(3:5,7:18)] <- rb_err
#     for (row in 1:length(temp$stats$pos)){
#       r <- res[[temp$stats$pos[row]]]
#       err <- sample(1:length(r[,1]), 1)
#       temp$stats[row,c(3:5,7:18)] <- temp$stats[row, c(3:5,7:18)] + r[err,]
#     }
    temp <- calcPoints(temp)
    temp <- buildvalues(temp)
    sims[,paste('values.',i,sep='')] <- temp$stats$values[match(sims[,1], temp$stats$PlayerName)]
#   sims <- merge(sims, temp$stats[,c("PlayerName","values")], by="PlayerName",
#                 suffixes=c('',paste('.',i,sep='')))
  }
  league[['sims']] <- sims
  return(league)
}

# FAIR WARNING: each iteration of my function takes ~1.05 seconds to perform.
## If you run this with nsim=10000, it's going to take ~2.92 hours to perform.
### I don't know how to make this shorter. I took out a for loop and the merge function
#### (it used to take longer), but other than those, I don't know how to make it run any faster...

## Question 2
quantile.league <- function(league, probs=c(0.25,0.5,0.75)) {
  if (!("sims" %in% names(l))){
    stop('ERROR... simulated dollar values not present in league.')
  }
  temp <- apply(league$sims[,-c(1,2)], 1, quantile, probs)
  quant <- data.frame(PlayerName=league$stats$PlayerName,Position=league$stats$pos)
  for (i in 1:length(probs)){
    quant <- cbind(quant,temp[i,])
    names(quant)[i+2] <- paste((probs[i]*100),'%',sep='')
  }
  return(quant)
}

## Question 3

conf.interval <- function(league, probs=c(0.25,0.5,0.75)){
  if (!("sims" %in% names(l))){
    stop('ERROR... simulated dollar values not present in league.')
  }
  temp <- quantile(league,probs)
  kicker <- temp[temp$Position == 'k',]
  kicker <- kicker[order(-kicker[,ncol(kicker)]), ]
  kicker <- kicker[1:(league$posReq$k * league$nTeams),]
  row.names(kicker) <- NULL
  quarter <- temp[temp$Position == 'qb',]
  quarter <- quarter[order(-quarter[,ncol(quarter)]), ]
  quarter <- quarter[1:(league$posReq$qb * league$nTeams),]
```

```
  row.names(quarter) <- NULL
  running <- temp[temp$Position == 'rb',]
  running <- running[order(-running[,ncol(running)]), ]
  running <- running[1:(league$posReq$rb * league$nTeams),]
  row.names(running) <- NULL
  wide <- temp[temp$Position == 'wr',]
  wide <- wide[order(-wide[,ncol(wide)]), ]
  wide <- wide[1:(league$posReq$wr * league$nTeams),]
  row.names(wide) <- NULL
  tight <- temp[temp$Position == 'te',]
  tight <- tight[order(-tight[,ncol(tight)]), ]
  tight <- tight[1:(league$posReq$te * league$nTeams),]
  row.names(tight) <- NULL
  conf <- list(qb =quarter, rb =running, wr=wide, te=tight, k=kicker)
  class(conf) <- 'league.conf.interval'
  return(conf)
}

## Question 4

plot.league.conf.interval <- function(conf, pos){
  len <- nrow(conf[[pos]][1])
  plot(1:len,t(conf[[pos]][3]), ylab="Dollar Value", xlab="Ranking", typ='l',
       ylim=c(0,round(max(conf[[pos]][,ncol(conf[[pos]])]))+2))
  leg <- c(1)
  for (i in 4:length(names(conf[[pos]]))){
    lines(1:len, t(conf[[pos]][i]), lty=(i-2))
    leg <- c(leg,i-2)
  }
  legend("topright", legend = names(conf[[pos]])[3:ncol(conf[[pos]])], lty=leg)
}
```

I will test your code with the following:

```
l1 <- addNoise(l, football, 500) # changed this for your computer's benefit
quantile(l1)
ci <- conf.interval(l1)
plot(ci, 'qb')
plot(ci, 'rb')
plot(ci, 'wr')
plot(ci, 'te')
plot(ci, 'k')
```