# Basketball

*Hannah Koschmeder*

*7/16/2018*

## Reading in the Data

```
event_codes <- read.delim("2018 Business & Basketball Analytics Hackathon Files/Basketball Analytics/NB
lineup <- read.delim("2018 Business & Basketball Analytics Hackathon Files/Basketball Analytics/NBA Hac
play_by_play <- read.delim("2018 Business & Basketball Analytics Hackathon Files/Basketball Analytics/N
```

## Loading Useful Libraries

```
library(plyr)
library(dplyr)
library(data.table)
```

## Task Basketball Analytics: Use the attached data to calculate plus/minus for each player in each game

```
newfn <- function(game_id, period) {
  players <- lineup %>%
    filter(Game_id == game_id, Period == period) %>%
    select(Person_id)
  players <- transpose(players)
  return(players)}
#add which players are in the starting lineup to every play
intense_play <- ddply(play_by_play,.(Game_id, Period),transform,p1=newfn(Game_id, Period)[1], p2=newfn(
      p7=newfn(Game_id, Period)[7], p8=newfn(Game_id, Period)[8], p9=newfn(Game_id, Period)[9],
      p10=newfn(Game_id, Period)[10])
```

## Discovering the Original Row ID is Equivalent to the Correct Sorting

```
#add a column with a more absolute sort order for each game
intense_play <- intense_play %>%
  arrange(Game_id, Period, desc(PC_Time), WC_Time, Event_Num) %>%
  tibble::rowid_to_column()
```

## Reflecting Substitutions in Lineup

```
#if a substitution is made, change the two players' ids for every row in the rest of the period of that

substitutions <- intense_play[intense_play[,"Event_Msg_Type"]==8,]

free_throw_sub <- function(rowidd){
```

```
  temp_rowid <- rowidd
  while(intense_play$Event_Msg_Type[temp_rowid-1]==8) temp_rowid <- temp_rowid-1
  if(intense_play$Event_Msg_Type[temp_rowid-1]==3){
  while(intense_play[intense_play$rowid==rowidd,"Event_Msg_Type"]==8|intense_play[intense_play$rowid==r
  return(rowidd)
}

for (substitution in seq(dim(substitutions)[1])){
  subs <- slice(substitutions,substitution)
  game_id = subs[["Game_id"]]
  period = subs[["Period"]]
  rowID = subs[["rowid"]]
  person1 = subs[["Person1"]]
  person2 = subs[["Person2"]]
  colname=names(subs)[match(person1, subs)]
  rowID = free_throw_sub(rowID)
  ind <- which(intense_play$rowid>=rowID & intense_play$Period==period & intense_play$Game_id==game_id)
  intense_play[ind,colname] <- as.character(person2)
  ind2 <- which(substitutions$rowid>=rowID & substitutions$Period==period & substitutions$Game_id==game_
  substitutions[ind2,colname] <- as.character(person2)
}
```

## Matching Players with their Teams

```
#add a column to intense_play to merge player1 accurately with their team
intense_play <- merge(intense_play, unique(lineup[,c("Game_id", "Person_id", "Team_id")]), by.x = c("Ga

#find which players were unable to be merged with their team...because they weren't in the initial line
team_missing_persons <- intense_play[is.na(intense_play$Team_id.y),c("Game_id","Person1", "Event_Msg_Ty

#match each missing person with the team of the person that they subbed in for during that game
funn <- function(x){
  game_id <- x[1]
  person <- x[2]
  matches <- which(intense_play$Game_id==game_id & intense_play$Person2==person & !is.na(intense_play$T
  if(length(matches)>0) return(unique(intense_play[matches,"Team_id.y"])) else return(factor(NA))
}

found_teams <- data.frame(Found = apply(team_missing_persons,1,funn))
found_teams <- tibble::rownames_to_column(found_teams)
found_teams <- found_teams[!is.na(found_teams$Found),]

#assign the found teams
intense_play[found_teams$rowname,"Team_id.y"] <- found_teams$Found
substitutions <- intense_play[intense_play[,"Event_Msg_Type"]==8,]

#restrict intense_play to point making plays
point_plays<- intense_play[which(intense_play$Event_Msg_Type==1|(intense_play$Event_Msg_Type==3 & intens

#showing that there are no more NAs for person1s team...the person scoring
sum(is.na(point_plays$Team_id.y))
```

```
## [1] 0
```

### Creating a full list of all player/game combinations

```
#put all players and games in one place
plusminus <- bind_rows(
  substitutions[, c("Game_id", "Person1", "Team_id.y")] %>%
    distinct(Game_id, Person1, .keep_all=T) %>%
    dplyr::rename(Person_id = Person1, Team_id=Team_id.y),
  substitutions[, c("Game_id", "Person2", "Team_id.y")] %>%
    distinct(Game_id, Person2, .keep_all=T) %>%
     dplyr::rename(Person_id = Person2, Team_id=Team_id.y),
  lineup[,c("Game_id", "Person_id", "Team_id")] %>%
    distinct(Game_id, Person_id, .keep_all=T)
  )[,c("Game_id", "Person_id", "Team_id")]

#make sure it is a distinct data frame of players and games
#plusminus[!is.na(plusminus$Team_id)]
plusminus <- distinct(plusminus, Game_id, Person_id,.keep_all=T)

#add each player's plus/minus column
plusminus[,"PM"] = 0
plusminus <- plusminus[,c("Game_id", "Person_id", "Team_id")]
```

### Calculating Plus/Minus for each player and game

```
match_points <- function(Game_id, Person_id, Team_id){
  relevant_rows <- point_plays[which(point_plays$Game_id==Game_id&
                       (point_plays$V1==Person_id|
                          point_plays$V2==Person_id|
                          point_plays$V3==Person_id|
                          point_plays$V4==Person_id|
                          point_plays$V5==Person_id|
                          point_plays$V6==Person_id|
                          point_plays$V7==Person_id|
                          point_plays$V8==Person_id|
                          point_plays$V9==Person_id|
                          point_plays$V10==Person_id)),]
  plus <- relevant_rows[which(relevant_rows$Team_id.y == Team_id), "Option1"]
  plus <- sum(plus)
  minus <- relevant_rows[which(relevant_rows$Team_id.y != Team_id), "Option1"]
  minus <- sum(minus)
  return(plus-minus)
}

plusminus <- mdply(plusminus, match_points)
plusminus <- rename(plusminus, Points=V1)
plusminus <- plusminus[,c("Game_id", "Person_id", "Points")]
```