Appendix A: Key R Code

Group 9

12/10/2018

```r
# Data Pre-Processing
library(devtools)
#devtools::install_github("abresler/nbastatR")
library(nbastatR)
library(plyr)
library(DataCombine)
setwd("~/Desktop/5291/5291_project")
#add team ranking
rank = read.csv("team_ranking_fornba.csv", header = F)
colnames(rank) = c("Ranking", "Long_Team", "Team")

#schedule 18-19
schedule_19_raw <- current_schedule()
schedule_19_regular <- schedule_19_raw[-(1:79),]
schedule_19 <- data.frame(as.character(schedule_19_regular$dateGame),
                          schedule_19_regular$slugTeamAway,
                          schedule_19_regular$slugTeamHome)
colnames(schedule_19) <- c("date", "teamaway","teamhome")

#team abbreviation in nba
nba_team_abbr <- function(team){
  team <- revalue(team, c("GS" = "GSW", "SA"="SAS", "NO"="NOP","NY"="NYK","PHO"="PHX"))
}

#gamelog 18-19:1230 games
gamelog_2019_full <- game_logs(seasons = 2019, league = "NBA",result_types = "player",
                               season_types = "Regular Season",nest_data = F,
                               assign_to_environment = TRUE, return_message = TRUE)
today <- min(which(gamelog_2019_full$dateGame=="2018-12-06"))
gamelog_2019_full <- gamelog_2019_full[c(1:(today-1)),]

schedule_19_tonight <- schedule_19[schedule_19$date == "2018-12-06",]

#import DK salary data
salary <- read.csv("DKSalaries1206.csv", header = T)
pre_game_data <- data.frame(salary$Name,salary$Position,
                            salary$TeamAbbrev, salary$Salary)
colnames(pre_game_data) <- c("name","position","team","salary")

# 1206
del <- which(pre_game_data$name == c("Devin Booker")|
               pre_game_data$name==c("Trey Burke")|
               pre_game_data$name == c("Carmelo Anthony")|
               pre_game_data$name==c("Kristaps Porzingis")|
               pre_game_data$name == c("Lance Thomas")|
               pre_game_data$name==c("Jabari Bird")|
               pre_game_data$name == c("Devin Booker")|
               pre_game_data$name==c("Trey Burke")|
               pre_game_data$name == c("CJ McCollum")|
               pre_game_data$name==c("T.J. Warren")|
               pre_game_data$name == c("Caleb Swanigan")|
               pre_game_data$name==c("George King")|
               pre_game_data$name == c("Zhou Qi")|
               pre_game_data$name==c("Brandon Knight"))


pre_game_data <- pre_game_data[-del,]

#add month
pre_game_data$Month <- 12

#add nba player id
pre_game_data$id <- c()
for (i in 1:length(pre_game_data$name)){
  pre_game_data$id[i] <- ifelse(length(nba_player_ids(players=pre_game_data$name[i]))!=
                                  0,nba_player_ids(players=pre_game_data$name[i]), NA)
}

sum(is.na(pre_game_data$id))

pre_game_data$name <- as.vector(pre_game_data$name)
pre_game_data$name[which(is.na(pre_game_data$id))]

nba_player_name <- function(name){
  name <- revalue(name, c("C.J. Miles"="CJ Miles",
                          "Jacob Evans III"="Jacob Evans",
                          "Luc Richard Mbah a Moute"="Luc Mbah a Moute",
                          "Frank Mason III"="Frank Mason",
                          "J.R. Smith"="JR Smith",
                          "Bruce Brown Jr."="Bruce Brown",
                          "P.J. Tucker"="PJ Tucker",
                          "Nazareth Mitrou-Long"="Naz Mitrou-Long",
                          "Mohamed Bamba"="Mo Bamba",
                          "Moe Harkless"="Maurice Harkless",
                          "D.J. Stephens"="DJ Stephens",
                          "Guillermo Hernangomez"="Willy Hernangomez",
                          "Sviatoslav Mykhailiuk"="Svi Mykhailiuk",
                          "Wayne Selden Jr."="Wayne Selden",
                          "Nene Hilario"="Nene"
                          ))
}

pre_game_data$name <- nba_player_name(pre_game_data$name)

pre_game_data$id <- c()
for (i in 1:length(pre_game_data$name)){
  pre_game_data$id[i] <- ifelse(length(nba_player_ids(players=pre_game_data$name[i]))!=
                                  0, nba_player_ids(players=pre_game_data$name[i]), NA)
}

pre_game_data$id[pre_game_data$id==961] <- 202322
sum(is.na(pre_game_data$id))

id_new <- pre_game_data$id

#add team and oppo name; home_Away
pre_game_data$team <- nba_team_abbr(pre_game_data$team)
```

```r
pre_game_data$opponent <- schedule_19_tonight$teamhome[
  match(pre_game_data$team, schedule_19_tonight$teamaway)]

for (i in 1:nrow(pre_game_data)){
  if(is.na(pre_game_data$opponent[i])){
    pre_game_data$opponent[i] <- schedule_19_tonight$teamaway[
                                  match(pre_game_data$team[i],
                                        schedule_19_tonight$teamhome)]
    pre_game_data$home_Away[i] <- c("H")
  }
  else{
    pre_game_data$opponent[i] <- schedule_19_tonight$teamhome[
                                  match(pre_game_data$team[i],
                                        schedule_19_tonight$teamaway)]
    pre_game_data$home_Away[i] <- c("A")
  }
}

#add team ranking
pre_game_data$team_ranking = rank$Ranking[match(pre_game_data$team, rank$Team)]

#add opponent ranking
pre_game_data$opponent_ranking = rank$Ranking[match(pre_game_data$opponent, rank$Team)]

player_bref_2018 <- bref_players_stats(seasons = 2018, tables = c("per_game"),only_total
s = F)
player_bref_2017 <- bref_players_stats(seasons = 2017, tables = c("per_game"),only_total
s = F)
player_bref_2016 <- bref_players_stats(seasons = 2016, tables = c("per_game"),only_total
s = F)

game_log_new <- c()
for (i in 1:length(id_new)){
  add <- subset(gamelog_2019_full, idPlayer == id_new[i])
  game_log_new <- rbind(game_log_new,add)
}
game_log_new_beta <-  data.frame(game_log_new$namePlayer, game_log_new$idPlayer,
                          game_log_new$dateGame, game_log_new$idGame, game_log_new$slugT
eam,
                          game_log_new$locationGame, game_log_new$slugOpponent,
                          game_log_new$outcomeGame,game_log_new$pts,game_log_new$fg3m,
                          game_log_new$treb,game_log_new$ast,game_log_new$stl,
                          game_log_new$blk,game_log_new$tov)
colnames(game_log_new_beta)<- c("namePlayer","idPlayer","dateGame","idGame","Team","home
_Away",
                                "Opponent","outcome","pts","3pm","reb","ast","stl","blk"
,"tov")

# Game Lag variables
game_log_new_final = list()
unique_player_new = unique(game_log_new_beta$idPlayer)
lag_var_new = c("pts","3pm","reb","ast","stl","blk","tov","outcome")
player_df = c()
```

```r
for (i in 1:length(unique_player_new)) {
  player_df = game_log_new_beta[game_log_new_beta$idPlayer == unique_player_new[i], ]
      for (j in 1:min(2, nrow(player_df))) {
            for (v in lag_var_new) {
                  player_df = slide(player_df, Var = v, slideBy = -j)
            }
      }
      game_log_new_final[[i]] = player_df[nrow(player_df),]
}

game_log_new_final <- rbind.fill(game_log_new_final)


# add pts season lag
pre_game_data$pts_season_lag1 <- player_bref_2018$ptsPerGame[
                                match(pre_game_data$id,player_bref_2018$idPlayerNBA)]

pre_game_data$pts_season_lag2 <- player_bref_2017$ptsPerGame[
                                match(pre_game_data$id,player_bref_2017$idPlayerNBA)]

pre_game_data$pts_season_lag3 <- player_bref_2016$ptsPerGame[
                                match(pre_game_data$id,player_bref_2016$idPlayerNBA)]

# add 3pm season lag
pre_game_data[,'X3pm_season_lag1'] <- player_bref_2018$fg3mPerGame[match(pre_game_data$i
d,player_bref_2018$idPlayerNBA)]

pre_game_data[,'X3pm_season_lag2'] <- player_bref_2017$fg3mPerGame[match(pre_game_data$i
d,player_bref_2017$idPlayerNBA)]

pre_game_data[,'X3pm_season_lag3'] <- player_bref_2016$fg3mPerGame[match(pre_game_data$i
d,player_bref_2016$idPlayerNBA)]

# add reb season lag
pre_game_data[,'reb_season_lag1'] <- player_bref_2018$trbPerGame[match(pre_game_data$id,
player_bref_2018$idPlayerNBA)]

pre_game_data[,'reb_season_lag2'] <- player_bref_2017$trbPerGame[match(pre_game_data$id,
player_bref_2017$idPlayerNBA)]

pre_game_data[,'reb_season_lag3'] <- player_bref_2016$trbPerGame[match(pre_game_data$id,
player_bref_2016$idPlayerNBA)]

# add ast season lag
pre_game_data[,'ast_season_lag1'] <- player_bref_2018$astPerGame[match(pre_game_data$id,
player_bref_2018$idPlayerNBA)]

pre_game_data[,'ast_season_lag2'] <- player_bref_2017$astPerGame[match(pre_game_data$id,
player_bref_2017$idPlayerNBA)]

pre_game_data[,'ast_season_lag3'] <- player_bref_2016$astPerGame[match(pre_game_data$id,
player_bref_2016$idPlayerNBA)]
```

```
# add stl season lag
pre_game_data[,'stl_season_lag1'] <- player_bref_2018$stlPerGame[match(pre_game_data$id,
player_bref_2018$idPlayerNBA)]

pre_game_data[,'stl_season_lag2'] <- player_bref_2017$stlPerGame[match(pre_game_data$id,
player_bref_2017$idPlayerNBA)]

pre_game_data[,'stl_season_lag3'] <- player_bref_2016$stlPerGame[match(pre_game_data$id,
player_bref_2016$idPlayerNBA)]

# add blk season lag
pre_game_data[,'blk_season_lag1'] <- player_bref_2018$blkPerGame[match(pre_game_data$id,
player_bref_2018$idPlayerNBA)]

pre_game_data[,'blk_season_lag2'] <- player_bref_2017$blkPerGame[match(pre_game_data$id,
player_bref_2017$idPlayerNBA)]

pre_game_data[,'blk_season_lag3'] <- player_bref_2016$blkPerGame[match(pre_game_data$id,
player_bref_2016$idPlayerNBA)]

# add tov season lag
pre_game_data[,'tov_season_lag1'] <- player_bref_2018$tovPerGame[match(pre_game_data$id,
player_bref_2018$idPlayerNBA)]

pre_game_data[,'tov_season_lag2'] <- player_bref_2017$tovPerGame[match(pre_game_data$id,
player_bref_2017$idPlayerNBA)]

pre_game_data[,'tov_season_lag3'] <- player_bref_2016$tovPerGame[match(pre_game_data$id,
player_bref_2016$idPlayerNBA)]


###### data for model fitting
##split position in to sub-position
library(base)
pos <- as.character(pre_game_data$position)
subp <- strsplit(pos, split="/")
rowpos <- lapply(subp, function(x){return(length(x))})
rowpos <- unlist(rowpos)
mulpos <- which(rowpos>1)

for(i in 1:length(mulpos)){
  pre_game_data$position[mulpos[i]] <- subp[[mulpos[i]]][1]
  new_row <- pre_game_data[mulpos[i], ]
  new_row$position <- subp[[mulpos[i]]][2]
  pre_game_data <- rbind(pre_game_data, new_row)
}

#
pre_game_data$pts.1 <- game_log_new_final$pts[match(pre_game_data$id,game_log_new_final$
idPlayer)]
pre_game_data$X3pm.1 <- game_log_new_final$`3pm`[match(pre_game_data$id,game_log_new_fin
al$idPlayer)]
pre_game_data$reb.1 <- game_log_new_final$reb[match(pre_game_data$id,game_log_new_final$
idPlayer)]
```

```
pre_game_data$ast.1 <- game_log_new_final$ast[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$stl.1 <- game_log_new_final$stl[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$blk.1 <- game_log_new_final$blk[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$tov.1 <- game_log_new_final$tov[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$outcome.1 <- game_log_new_final$outcome[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$pts.2 <- game_log_new_final$`pts-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$X3pm.2 <- game_log_new_final$`3pm-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$reb.2 <- game_log_new_final$`reb-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$ast.2 <- game_log_new_final$`ast-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$stl.2 <- game_log_new_final$`stl-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$blk.2 <- game_log_new_final$`blk-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$tov.2 <- game_log_new_final$`tov-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$outcome.2 <- game_log_new_final$`outcome-1`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$pts.3 <- game_log_new_final$`pts-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$X3pm.3 <- game_log_new_final$`3pm-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$reb.3 <- game_log_new_final$`reb-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$ast.3 <- game_log_new_final$`ast-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$stl.3 <- game_log_new_final$`stl-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$blk.3 <- game_log_new_final$`blk-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$tov.3 <- game_log_new_final$`tov-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]
pre_game_data$outcome.3 <- game_log_new_final$`outcome-2`[match(pre_game_data$id,game_log_new_final$idPlayer)]

#write.csv(pre_game_data,"pre_game_data_1206.csv")
```

```
#GBM Tuning Code:

library(gbm)

 tune_gbm=function(t,i,s,n,b,k,dataset)
  {
    row=1
    mat=matrix(NA,ncol=7,nrow=length(i)*length(s)*length(n)*length(b)*length(k))
    colnames(mat)=c("trees","interdepth","shrinkage","n.minobsinnode",
                    "bag.fraction","k","error")
    for (inter in i)
    {
      for (shr in s)
      {
        for (n.m in n)
        {
          for (bf in b)
          {
            for (K in k)

            {
              set.seed(1)
            gbm=gbm(score~., verbose=TRUE, distribution="laplace",data=dataset, n.trees=
t,
                    interaction.depth=inter, shrinkage=shr, n.minobsinnode=n.m,
                    bag.fraction=bf, cv.folds=K)
            error=min(gbm$cv.error)
            mat[row,]=c(which.min(gbm$cv.error),inter,shr,n.m,bf,K,error)
            row=row+1
              }
          }
        }
      }
    }

    return(mat)
  }

Sample GBM code:
tuning=tune_gbm(t=2000, i=seq(from=3,to=6,by=1), s=0.1, n=7,
                b=seq(from=0.3,to=0.5,by=0.1), k=5, dataset=train_full)
```

```
#XGBoost Tuning Code:

library(Matrix)
library(xgboost)

xgboost_tuning = function(n, b, e, g, d, child, s, c, t, l, lb, a, r, K, early, data)
{
  start_time = Sys.time()
  set.seed(1)
  samp = sample(1:nrow(data), nrow(data))
  row = 1
  combination = 1
  mat=matrix(NA,ncol=14,nrow=length(b)*length(e)*length(g)*length(d)*
               length(child)*length(s)*length(c)*length(t)*length(l)*
               length(lb)*length(a)*length(r))
  colnames(mat)=c("b","e","g","d", "ch","s", "c",  "t", "l","lb","a", "r","early", "erro
r")
  for (b_curr in b)
  {
    for (e_curr in e)
    {
      for (g_curr in g)
      {
        for (d_curr in d)
        {
          for (child_curr in child)
          {
            for (s_curr in s)
            {
              for (c_curr in c)
              {
                for (t_curr in t)
                {
                  for (l_curr in l)
                  {
                    for (lb_curr in lb)
                    {
                      for (a_curr in a)
                      {
                        for (r_curr in r)
                        {
                          errors = c()
                          time1 = Sys.time()
                          for (k in 1:K)
                          {
                            train_indices = samp[((k-1)/5*length(samp)+1):
                                                   ((k/5)*length(samp))]
                            train = data[train_indices,]
                            test = data[-train_indices,]

                            options(na.action = "na.pass")
                            train_sparse = sparse.model.matrix(score~., data = train)
                            train_model_mat = xgb.DMatrix(data = as.matrix(train_sparse)
,
                                                          label = train$score)

                            options(na.action = "na.pass")
                            test_sparse = sparse.model.matrix(score~., data = test)
                            test_model_mat = xgb.DMatrix(data = as.matrix(test_sparse),
                                                         label = test$score)

                            params = list(booster =  b_curr, eta = e_curr, gamma = g_cur
r,
                                          max_depth = d_curr, min_child_weight = child_c
urr,
                                          subsample = s_curr, colsample_bytree = c_curr,

                                          num_parallel_tree = t_curr, lambda = l_curr,
                                          lambda_bias = lb_curr, alpha = a_curr, reg = r
_curr)

                            evalerror <- function(preds, train_model_mat) {
                              labels <- getinfo(train_model_mat, "label")
                              err <- median(abs(labels-preds))
                              return(list(metric = "error", value = err))
                            }
                            set.seed(1)
                            fit_xgb <- xgb.train(data = train_model_mat,
                                                 nrounds = n,
                                                 params = params,
                                                 watchlist = list(train = train_model_ma
t,
                                                                  test = test_model_mat)
,
                                                 verbose = 0,
                                                 early_stopping_rounds = early,
                                                 maximize = FALSE,
                                                 feval = evalerror)
                            best_iter=which.min(fit_xgb$evaluation_log$test_error)
                            print(best_iter)
                            errors = c(errors,min(fit_xgb$evaluation_log$test_error))

                          }
                          time2 = Sys.time()
                          mat[row,]=c(b_curr, e_curr, g_curr, d_curr, child_curr,
                                      s_curr, c_curr, t_curr, l_curr, lb_curr,
                                      a_curr, r_curr, early, mean(errors))

                          row = row + 1
                          print(paste("Took",difftime(time2, time1, units = "mins"),
                                      "minutes"))
                          print(paste("Finished tuning parameter combination number",
                                      combination, "of", nrow(mat)))
                          combination = combination + 1
                        }
                      }
                    }
```

```
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  end_time = Sys.time()
  print(paste("Entire tuning took",difftime(end_time,start_time,units="mins"),
              "minutes"))
  return(mat)
}
```

```
# Sample GMB code
tuning=tune_gbm(t=2000, i=seq(from=3,to=6,by=1), s=0.1, n=7,
                b=seq(from=0.3,to=0.5,by=0.1, k=5, dataset=train_full))
```

```
# Sample Xgboost Code:

xgboost_tuning(n = 100, b = "gbtree", e = 0.1, g = 0.01, d = 3,
               child = 5, s = 0.7, c = 0.5,
               t = 2, l = 0, lb = 0, a = 0,
               r = "reg:linear", K = 5, early = 20, model_data)
```

```r
# Sample Prediction code (Example of Dec 6th):

library(dplyr)
library(gbm)
set.seed(1)

#subset model data into positions:
AllData <- read.csv("final_model_data.csv")

SG_dat<- AllData[,-1] %>% filter(position == "SG") %>% select(-position)
PG_dat<- AllData[,-1] %>% filter(position == "PG") %>% select(-position)

PF_dat<- AllData[,-1] %>% filter(position == "PF") %>% select(-position)
SF_dat<- AllData[,-1] %>% filter(position == "SF") %>% select(-position)

C_dat<- AllData[,-1] %>% filter(position == "C") %>% select(-position)

#subset current data into positions:
#pre_game_data <- read.csv("pre_game_data_1206.csv")
model_sg_data=pre_game_data[pre_game_data$position=="SG",][,-c(1:4,6,7)]
model_pg_data=pre_game_data[pre_game_data$position=="PG",][,-c(1:4,6,7)]

model_pf_data=pre_game_data[pre_game_data$position=="PF",][,-c(1:4,6,7)]
model_sf_data=pre_game_data[pre_game_data$position=="SF",][,-c(1:4,6,7)]

model_c_data=pre_game_data[pre_game_data$position=="C",][,-c(1:4,6,7)]

model_utl_data=pre_game_data[-c(1:4,6,7)]

# Model Prediction for each position

#Model prediction SG
SG_model <- gbm(score~., verbose=TRUE, distribution="laplace", data=SG_dat,
                n.trees=765, interaction.depth=8, shrinkage=0.01,
                n.minobsinnode=20, bag.fraction=0.35)
SG_fit <- predict(SG_model, model_sg_data, n.trees = 765)

#Model prediction PG
PG_model <- gbm(score~., verbose=TRUE, distribution="laplace", data=PG_dat,
                n.trees=61, interaction.depth=8, shrinkage=0.08,
                n.minobsinnode=20, bag.fraction=0.60)
PG_fit <- predict(PG_model, model_pg_data, n.trees = 61)

#Model prediction SF
SF_model <- gbm(score~., verbose=TRUE, distribution="laplace", data=SF_dat,
                n.trees=733, interaction.depth=9, shrinkage=0.01,
                n.minobsinnode=6, bag.fraction=0.46)
SF_fit <- predict(SF_model, model_sf_data, n.trees = 733)

#Model prediction PF
PF_model <- gbm(score~., verbose=TRUE, distribution="laplace", data=PF_dat,
                n.trees=40, interaction.depth=10, shrinkage=0.01,
                n.minobsinnode=14, bag.fraction=0.70)
```

```
PF_fit <- predict(PF_model, model_pf_data, n.trees = 40)

#Model prediction C
C_model <- gbm(score~., verbose=TRUE, distribution="laplace", data=C_dat,
               n.trees=309, interaction.depth=7, shrinkage=0.02,
               n.minobsinnode=7, bag.fraction=0.70)
C_fit <- predict(C_model, model_c_data, n.trees = 309)


#SG
SG_predict <- data.frame(pre_game_data[pre_game_data$position=="SG",]$name,
                         pre_game_data[pre_game_data$position=="SG",]$position,
                         pre_game_data[pre_game_data$position=="SG",]$salary,
                         SG_fit)
colnames(SG_predict) <- c("name","position","salary","score")

#PG
PG_predict <- data.frame(pre_game_data[pre_game_data$position=="PG",]$name,
                         pre_game_data[pre_game_data$position=="PG",]$position,
                         pre_game_data[pre_game_data$position=="PG",]$salary,
                         PG_fit)
colnames(PG_predict) <- c("name","position","salary","score")

# G
G_predict <- rbind(SG_predict,PG_predict)

#SF
SF_predict <- data.frame(pre_game_data[pre_game_data$position=="SF",]$name,
                         pre_game_data[pre_game_data$position=="SF",]$position,
                         pre_game_data[pre_game_data$position=="SF",]$salary,
                         SF_fit)
colnames(SF_predict) <- c("name","position","salary","score")

#PF
PF_predict <- data.frame(pre_game_data[pre_game_data$position=="PF",]$name,
                         pre_game_data[pre_game_data$position=="PF",]$position,
                         pre_game_data[pre_game_data$position=="PF",]$salary,
                         PF_fit)
colnames(PF_predict) <- c("name","position","salary","score")

#F
F_predict <- rbind(SF_predict, PF_predict)

#C
C_predict <- data.frame(pre_game_data[pre_game_data$position=="C",]$name,
                         pre_game_data[pre_game_data$position=="C",]$position,
                         pre_game_data[pre_game_data$position=="C",]$salary,
                         C_fit)
colnames(C_predict) <- c("name","position","salary","score")

# Utility
UTL_predict <- rbind(G_predict,F_predict,C_predict)
```

```
# Calculation of DraftKings Fantasy Score and
# Simple Evaluation of individual player prediction results:

library(nbastatR)
#gamelog 18-19:1230 games
gamelog_2019_full <- game_logs(seasons = 2019, league = "NBA", result_types = "player",
                               season_types = "Regular Season",nest_data = F,
                               assign_to_environment = TRUE, return_message = TRUE)

game_log_today <- gamelog_2019_full[which(gamelog_2019_full$dateGame=="2018-12-06"),]

game_log_today <-  data.frame(game_log_today$namePlayer,game_log_today$idPlayer,
                            game_log_today$dateGame,game_log_today$pts,
                            game_log_today$fg3m,game_log_today$treb,
                            game_log_today$ast,game_log_today$stl,game_log_today$blk,
                            game_log_today$tov)
colnames(game_log_today)<- c("namePlayer","idPlayer","dateGame","pts",
                             "X3pm","reb","ast","stl","blk","tov")

#calculate double double and triple double
game_log_today$sum <- apply(game_log_today[,c(4,6:9)]>=10,1,sum)
game_log_today$double_double <- ifelse(game_log_today$sum == 2,1,0)
game_log_today$triple_double <- ifelse(game_log_today$sum == 3,1,0)

game_log_today$score <- game_log_today$pts+0.5*game_log_today$X3pm+
                        1.25*game_log_today$reb+1.5*game_log_today$ast+
                        2*game_log_today$stl+2*game_log_today$blk-0.5*game_log_today$tov
 +
                        1.5*game_log_today$double_double+3*game_log_today$triple_double

game_log_today<- data.frame(game_log_today$namePlayer,
                            game_log_today$dateGame,
                            game_log_today$score)

#write.csv(game_log_today,"game_log_1206.csv")

game_log_today$predicted <- UTL_predict$score[match(game_log_today$game_log_today.namePl
ayer,
                                                    UTL_predict$name)]

game_log_today <- game_log_today[-which(is.na(game_log_today$predicted)),]


#median absolute error:
median(abs(game_log_today$predicted-game_log_today$game_log_today.score))

#median percent error:
median(abs((game_log_today$predicted-game_log_today$game_log_today.score)/
             game_log_today$game_log_today.score))

#mean absolute error:
mean(abs(game_log_today$predicted-game_log_today$game_log_today.score))
```

```
#R Squared:
(cor(game_log_today$predicted,game_log_today$game_log_today.score))^2
```