ckme136\_project

Lorne Mach

April 6, 2018

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

# Load packages and functions

#install.packages("factoextra")  
#install.packages("dplyr")  
#install.packages("DAAG")  
#install.packages("fpc")  
#install.packages("cluster")  
#install.packages("ggplot2")  
#install.packages("rlang")  
#install.packages("dbscan")  
#install.packages("miscTools")  
#install.packages("forecast")  
  
library(rlang)

## Warning: package 'rlang' was built under R version 3.3.3

library(ggplot2) # Used by graph packages below

## Warning: package 'ggplot2' was built under R version 3.3.3

library(corrplot) # Used for correlation visualization

## Warning: package 'corrplot' was built under R version 3.3.3

library(plyr)

## Warning: package 'plyr' was built under R version 3.3.3

library(dplyr) # Used for group\_by function

## Warning: package 'dplyr' was built under R version 3.3.3

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(MASS) # Used for AIC functions

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

library(factoextra) # Used for cluster visualization and visualization of optimal cluster selection

## Warning: package 'factoextra' was built under R version 3.3.3

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

library(cluster) # Used for cluster analysis functions like clusGap and silhouette

## Warning: package 'cluster' was built under R version 3.3.3

library(dbscan)

## Warning: package 'dbscan' was built under R version 3.3.3

library(forecast) # Used for the accuracy() function to calculate various statistics rmse, rsquared, mae for regression model analysis

## Warning: package 'forecast' was built under R version 3.3.3

library(caret) #Used for leveraging cross fold validation for linear and random forest models

## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

# Input: teams dataframe, year to start searching for winning teams  
# Output: winning teams dataframe  
# Desc: Get winning teams starting from a specific year  
getWinningTeams <- function(teams,startYear) {  
 winningteams = teams[teams$year > startYear,]  
 winningteams = winningteams[winningteams$div\_win == "Y" |  
 winningteams$wc\_win == "Y" |   
 winningteams$lg\_win == "Y" |   
 winningteams$ws\_win == "Y",]  
   
   
 return(winningteams)  
}  
  
# Input: fielding dataframe  
# Output: fielding dataframe without NA and a single primary position for a players  
# Desc: Players can play multiple positions on a baseball team, this function finds the primary position and also removes NAs  
getFieldingClean <- function(fielding) {  
  
 # Drop all rows with NA in po or a column  
 fielding\_temp=fielding[!is.na(fielding$po) & !is.na(fielding$a),]  
   
 # Players can play multiple positions  
 # The primary position of the player is the position where he has accumulated the most po and a  
 fielding\_temp$po\_a <- rowSums(fielding\_temp[,c("po","a")])  
  
 fielding\_clean=fielding\_temp %>% group\_by(year,player\_id) %>% filter(po\_a == max(po\_a))  
  
 return(fielding\_clean)  
}  
  
# Input: All dataframes that contain data relevant to batting  
# Output: Batting metrics of players from winning teams starting from a specific year and with a minimum number of at-bats(ab)  
# Desc: Get batting metrics of players from winning teams starting from a specific year and with a minimum number of at-bats(ab)  
getWinningBatters <- function(teams,woba,batting,fielding,salary,usatodaySalary,player,allstar,award,startYear, minAb)  
{  
 winningTeams=getWinningTeams(teams,startYear)  
 winningBatters=merge(winningTeams[,c("year","team\_id")], batting, by=c("year","team\_id"))  
 winningBatters=getBatters(woba,winningBatters,fielding,salary,usatodaySalary,player,allstar,award,startYear,minAb)  
 return(winningBatters)  
}   
  
# Input: All dataframes that contain data relevant to batting  
# Output: Batting metrics of players from teams starting from a specific year and with a minimum number of at-bats(ab)  
# Desc: Get batting metrics of players from teams starting from a specific year and with a minimum number of at-bats(ab)  
getBatters <- function(woba,batting, fielding, salary, usatodaySalary,player, allstar,award,startYear, minAb)  
{  
 batters=merge(fielding[,c("player\_id","pos","year","po", "a","e")],batting,by=c("player\_id","year"))  
  
 batters=merge(salary[,c("year", "player\_id", "salary")], batters,by=c("player\_id","year"))  
 batters=merge(player[,c("player\_id","debut","birth\_year","name\_first","name\_last")],batters,by=c("player\_id"))  
 batters=merge(allstar[,c("player\_id","year","game\_id")],batters,by=c("player\_id","year"),all.y=TRUE)  
 batters=merge(award[,c("player\_id","year","award\_id")],batters,by=c("player\_id","year"),all.y=TRUE)  
 batters$isAllStar=!is.na(batters$game\_id)  
 batters$isAwardWinner=!is.na(batters$award\_id)  
 batters$debut=as.Date(batters$debut)  
 batters$hasFreeAgentStatus=batters$year - as.numeric(format(batters$debut,"%Y")) >= 6  
 batters$age=batters$year - as.numeric(batters$birth\_year)  
 batters$ageUnder25=(batters$year - as.numeric(batters$birth\_year) < 25)  
 batters$age25to30=(batters$year - as.numeric(batters$birth\_year) >=25) &   
 (batters$year - as.numeric(batters$birth\_year) < 30)  
 batters$age30to35=(batters$year - as.numeric(batters$birth\_year) >=30) &   
 (batters$year - as.numeric(batters$birth\_year) < 35)  
 batters$age35to50=(batters$year - as.numeric(batters$birth\_year) >=35)  
 batters=batters[which(batters$ab > minAb),]  
 batters=batters[which(batters$year > startYear),]  
 batters=batters[!duplicated(batters[c("player\_id","year")]),]  
  
 batters=merge(usatodaySalary[,c("Year","Team","FirstName","LastName","Avg.Annual","UTSalary")],batters,by.x=c("Year","Team","FirstName","LastName"),by.y=c("year","team\_id","name\_first","name\_last"),all.y=TRUE)  
  
 missingAvgAnnual=which(is.na(batters$Avg.Annual))  
 batters[missingAvgAnnual,]$Avg.Annual=batters[missingAvgAnnual,]$salary  
   
 batters = merge(batters,woba[,c("Season","wBB","wHBP","w1B","w2B","w3B","wHR")],by.x=c("Year"),by.y=c("Season"))  
 batters$single=batters$h - batters$double - batters$triple - batters$hr  
   
# woba statistic = (x1 \* BB + x2 \* HBP + x3 \* 1B + x4 \* 2B + x5 \* 3B + x6 \* HR)/(AB + BB + IBB + SF + HBP)  
# https://www.fangraphs.com/library/woba-as-a-gateway-statistic/  
   
 batters$woba=(batters$wBB \* batters$bb +  
 batters$wHBP \* batters$hbp +  
 batters$w1B \* batters$single +  
 batters$w2B \* batters$double +  
 batters$w3B \* batters$triple +  
 batters$wHR \* batters$hr) / (batters$ab + batters$bb - batters$ibb + batters$sf + batters$hbp)  
  
  
 return(batters)  
}  
  
# Input: dataframe, features  
# Output: PCA results  
# Desc: Perform PCA on a set of features from a dataframe  
performPCA <- function(object,features) {  
 result=princomp(object[,features],cor=T)  
 return(result)  
}  
  
# Input: dataframe and features  
# Output: plot of the correlation between features in a dataframe  
# Desc: Plot of the correlation between features in a dataframe  
showBattersCorPlot <- function(batters,features) {  
 batters\_corr=cor(batters[,features])  
 corrplot::corrplot(batters\_corr,method="pie")  
 return(batters\_corr)  
}  
  
# Input: dataframe and features  
# Desc: Show summary statistics and boxplots of features in a dataframe  
showSummaryStatistics <- function(batters, features) {  
 batters\_temp=batters  
 result=sapply(batters\_temp, function (x) if (is.numeric(x)) { 1 } else { 0 })  
 numeric\_features=names(which(result==1))  
 par(mfrow = c(2,3))  
 for (feature in numeric\_features)  
 {  
 boxplot(batters\_temp[,c(feature)],main=feature)  
 print(feature)  
 print(summary(batters\_temp[,c(feature)]))  
 }   
 par(mfrow = c(1,1))  
}  
  
  
# Input: dataframe and features  
# Output: dataframe with outliers removed  
# Desc: Removes outliers from a dataframe  
removeOutliers <- function(batters, features) {  
 batters\_temp=batters  
 for (feature in features)  
 {  
 result=summary(batters\_temp[,c(feature)])  
 q3=result[5]  
 q1=result[2]  
 iqr=q3-q1  
 upper\_fence=q3+(1.5\*iqr)  
 lower\_fence=q1-(1.5\*iqr)  
 # print(batters\_temp[which(batters\_temp[,c(feature)] > upper\_fence |   
 # batters\_temp[,c(feature)] < lower\_fence),])  
 batters\_temp=batters\_temp[which(batters\_temp[,c(feature)] < upper\_fence &   
 batters\_temp[,c(feature)] > lower\_fence),]  
 }  
 return(batters\_temp)  
}  
  
# Input: dataframe and features  
# Output: dataframe of outliers  
# Desc: Get outliers from a dataframe  
getOutliers <- function(batters, features) {  
 batters\_temp=batters  
 for (feature in features)  
 {  
 result=summary(batters\_temp[,c(feature)])  
 q3=result[5]  
 q1=result[2]  
 iqr=q3-q1  
 upper\_fence=q3+(1.5\*iqr)  
 lower\_fence=q1-(1.5\*iqr)  
 batters\_temp=batters\_temp[which(batters\_temp[,c(feature)] > upper\_fence |   
 batters\_temp[,c(feature)] < lower\_fence),]  
 }  
 return(batters\_temp)  
}  
  
# Input: dataframe and features  
# Output: dataframe with outliers removed and replaced with upper fence values  
# Desc: Outliers cause issues, instead of removing them, substitute the outliers with upper fence values  
  
substituteOutliersWithUpperFence <- function(batters, features) {  
 batters\_temp=batters  
 for (feature in features)  
 {  
 result=summary(batters\_temp[,c(feature)])  
 q3=result[5]  
 q1=result[2]  
 iqr=q3-q1  
 upper\_fence=q3+(1.5\*iqr)  
 lower\_fence=q1-(1.5\*iqr)  
 batters\_temp[which(batters\_temp[,c(feature)] > upper\_fence),c(feature)] = upper\_fence  
   
 }  
 return(batters\_temp)  
}  
  
# Input: Model, dataframe, string name of observed value, string name of model  
# Output: Returns dataframe with absolute deviation and percent error of each prediction made for every row in the dataframe  
# Desc: Find the absolute deviation and percent error of each prediction made for every row in the dataframe  
getModelPredictionError<-function(model,actual,modelName,battersTemp) {  
 salary.pred.modelName=paste("salary.pred.",modelName,sep="")  
 pred.err.percent.modelName=paste("pred.err.percent.",modelName,sep="")  
 salary.absdev.modelName=paste("salary.absdev.",modelName,sep="")  
   
 salary.predicted=predict(model,newdata=battersTemp)  
 battersTemp[,c(salary.pred.modelName)]=salary.predicted  
 battersTemp[,c(salary.absdev.modelName)]=abs(battersTemp[,c(actual)] - battersTemp[,c(salary.pred.modelName)])  
 battersTemp[,c(pred.err.percent.modelName)]=  
 abs(battersTemp[,c(salary.absdev.modelName)]/battersTemp[,c(actual)]) \* 100  
 return(battersTemp)  
}  
  
# Input: Log Model, dataframe, string name of observed value, string name of model  
# Output: Returns dataframe with absolute deviation and percent error of each prediction made for every row in the dataframe  
# Desc: Find the absolute deviation and percent error of each prediction made for every row in the dataframe  
getLogModelPredictionError<-function(model,actual,modelName,battersTemp) {  
 salary.pred.modelName=paste("salary.pred.",modelName,sep="")  
 pred.err.percent.modelName=paste("pred.err.percent.",modelName,sep="")  
 salary.absdev.modelName=paste("salary.absdev.",modelName,sep="")  
 pred.err.percent.inverse.log=paste("pred.err.percent.inverse.log",modelName,sep="")  
   
 salary.predicted=predict(model,newdata=battersTemp)  
 battersTemp[,c(salary.pred.modelName)]=salary.predicted  
 battersTemp[,c(salary.absdev.modelName)]=abs(log(battersTemp[,c(actual)]) - battersTemp[,c(salary.pred.modelName)])  
   
 battersTemp[,c(pred.err.percent.modelName)]=  
 abs(battersTemp[,c(salary.absdev.modelName)]/log(battersTemp[,c(actual)])) \* 100  
  
 inverse.log.absdev=abs(battersTemp[,c(actual)] - exp(battersTemp[,c(salary.pred.modelName)]))  
  
 battersTemp[,c(pred.err.percent.inverse.log)]=  
 abs(inverse.log.absdev/battersTemp[,c(actual)]) \* 100  
   
 return(battersTemp)  
}  
  
  
# Input: cluster data  
# Desc: Show plot of silhouette method of choosing clusters  
showAvgSilhouette <- function(clusterData) {  
 avg\_sil\_values=""  
 k.values <- 2:10  
 for (k in k.values)  
 {  
 km.res <- kmeans(clusterData, centers = k, nstart = 25)  
 ss <- silhouette(km.res$cluster, dist(clusterData))  
 avg\_sil\_values[k-1]=mean(ss[, 3])  
 }  
   
 print(plot(k.values, avg\_sil\_values,  
 type = "b", pch = 19, frame = FALSE,   
 xlab = "Number of clusters K",  
 ylab = "Average Silhouettes"))  
}  
  
  
# Input: dataframe and features  
# OUtput: standardized dataframe with mean 0 and standard deviation of 1  
# Desc: REturns a standardized dataframe  
getScaledClusterData<-function(batters, pos, features) {  
  
 result=batters  
 if (pos != "ALL")  
 {   
 result=batters[which(batters$pos==pos),features]  
 }  
 else  
 {  
 result=batters[which(batters$pos!="P"),features]  
 }   
  
 row.names(result)=paste(result$player\_id,result$Year,sep="")  
   
 result=scale(substituteOutliersWithUpperFence(result[,c(-1,-2)],names(result[,c(-1,-2)])))  
 return(result)  
}  
  
# Input: cluster data  
# Desc: Show plots of elbow, silhouette, and gap statistic to help determine best cluster selection  
showOptimalClusters<-function(clusterData)  
{  
 set.seed(123)  
  
 # Elbow method  
 print(fviz\_nbclust(clusterData, kmeans, method = "wss"))  
  
 # Silhoutte method  
 showAvgSilhouette(clusterData)  
  
 # Gap statistic  
 gap\_stat <- clusGap(clusterData, FUN = kmeans, nstart = 25,K.max = 10, B = 10)  
 print(fviz\_gap\_stat(gap\_stat))  
}  
  
# Desc: Show regression model statistics: "RMSE", "rsquared"","ME","RMSE","MAE","MPE","MAPE"  
modelSummary <- function (data, lev = NULL, model = NULL) {  
 mainStats <- postResample(data[, "pred"], data[, "obs"])  
 extraStats <- accuracy(data[, "pred"], data[, "obs"])  
 names(extraStats)=c("ME","RMSE","MAE","MPE","MAPE")  
 c(mainStats,extraStats)  
}  
  
visualModelOutliers<- function(model,pos,outlierFeature,salary.absdev.pred)  
{  
 model.outliers=getOutliers(model,outlierFeature)  
 freeagent=model.outliers[which(model.outliers$hasFreeAgentStatus==TRUE),salary.absdev.pred]  
nofa=model.outliers[which(model.outliers$hasFreeAgentStatus==FALSE),salary.absdev.pred]  
  
 View(model.outliers[which(model.outliers$hasFreeAgentStatus==TRUE),])  
 View(model.outliers[which(model.outliers$hasFreeAgentStatus==FALSE),])  
title=paste(pos," Position Salary Outliers (Actual-Predicted)",sep=" ")  
  
boxplot(list(freeagent,nofa),main=title,col=c("blue","green"),names=c("fa", "no-fa"),horizontal = TRUE)  
  
# model.no.outliers=removeOutliers(model,outlierFeature)  
  
 remove=row.names(model.outliers[which(model.outliers$hasFreeAgentStatus==TRUE),])  
 model.no.outliers=model[!rownames(model) %in% remove,]  
 return(model.no.outliers)  
}  
  
visualAbsoluteDev<- function(model,pos,salary.absdev.pred)  
{  
# model.outliers=getOutliers(model,err.percent)  
freeagent=model[which(model$hasFreeAgentStatus==TRUE),salary.absdev.pred]  
nofa=model[which(model$hasFreeAgentStatus==FALSE),salary.absdev.pred]  
   
title=paste(pos," Position Salary (Actual - Predicted)",sep=" ")  
  
boxplot(list(freeagent,nofa),main=title,col=c("blue","green"),names=c("fa", "no-fa"),horizontal = TRUE)  
  
}  
  
  
# Desc: Perform stepwise AIC to determine best features for linear regression, using the best features, create a model and run cross validation  
# Output: Error of the regression model produced  
buildAndRunLinearModel<-function(fullModel,nullModel,responseVar,modelName,modelData,modelCtrlLM)  
{  
 full=fullModel  
 null=nullModel  
 sboth=stepAIC(null,direction="both",scope=list(upper=full,lower=null),trace=FALSE)  
   
 modelFeatures=attr(sboth$terms,"term.labels")  
 print(modelFeatures)  
 model=train(modelData[,modelFeatures],modelData[,responseVar],method='lm',trControl=modelCtrlLM)  
   
 print(summary(model$finalModel))  
 print(model)  
   
# par(mfrow = c(2,2))  
 print(plot(model$finalModel))  
# par(mfrow = c(1,1))  
   
 error=getModelPredictionError(model,responseVar,modelName,modelData)  
 return(error)  
}  
  
# Desc: Perform stepwise AIC to determine best features for linear regression with natural log applied to response variable, using the best features, create a model and run cross validation  
# Output: Error of the regression model produced  
buildAndRunLogModel<-function(fullModel,nullModel,responseVar,modelName,modelData,modelCtrlLM)  
{  
 full=fullModel  
 null=nullModel  
 sboth=stepAIC(null,direction="both",scope=list(upper=full,lower=null),trace=FALSE)  
   
 modelFeatures=attr(sboth$terms,"term.labels")  
 print(modelFeatures)  
 model=train(modelData[,modelFeatures],log(modelData[,responseVar]),method='lm',trControl=modelCtrlLM)  
   
 print(summary(model$finalModel))  
 print(model)  
   
  
 print(plot(model$finalModel))  
  
 error=getLogModelPredictionError(model,responseVar,modelName,modelData)  
 return(error)  
}  
  
  
# Desc: Perform recursive feature elimination to determine best features for random forest regression, using the best features, create a model and run cross validation  
# Output: Error of the regression model produced  
buildAndRunRfModel<-function(responseVar,modelName,modelData,modelCtrlRF)  
{  
 control=rfeControl(functions=rfFuncs, method="cv", number=3)  
 result=rfe(modelData[,c(-1)],modelData[,responseVar],rfeControl=control)  
# plot(result, type=c("g", "o"))  
 result$variables[which(result$variables[,3]==16 & result$variables[,4]=="Fold1"),"var"]  
   
 result$variables[which(result$variables[,3]==16 & result$variables[,4]=="Fold2"),"var"]  
   
 result$variables[which(result$variables[,3]==16 & result$variables[,4]=="Fold3"),"var"]  
  
 modelFeatures=result$variables[which(result$variables[,3]==16 & result$variables[,4]=="Fold3"),"var"]  
  
 model=train(modelData[,modelFeatures],modelData[,responseVar],method='rf', trControl=modelCtrlRF)  
   
 print(modelFeatures)  
 print(model)  
   
 error=getModelPredictionError(model,responseVar,modelName,modelData)  
 return(error)  
   
}   
  
  
# Load datasets into dataframes  
teams = read.csv("team.csv")  
batting = read.csv("batting.csv")  
pitching = read.csv("pitching.csv")  
fielding = read.csv("fielding.csv")  
salary = read.csv("salary.csv")  
usatodaySalary = read.csv("usatoday\_salary2.csv")  
player = read.csv("player.csv")  
allstar = read.csv("all\_star.csv")  
award = read.csv("player\_award.csv")  
award = award[!duplicated(award[c("player\_id","year")]),]  
woba = read.csv("woba.csv")  
  
fieldingClean=getFieldingClean(fielding)

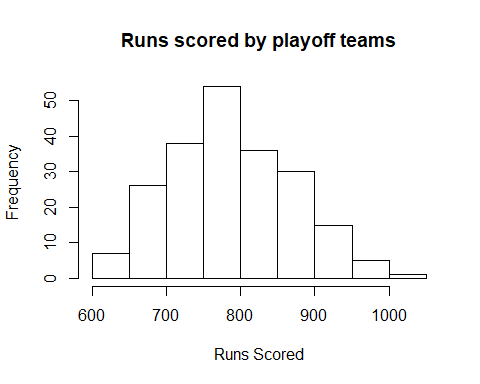
## Warning: package 'bindrcpp' was built under R version 3.3.3

# Get winning (playoff) teams from 1985-2015 and show their run scoring metrics

playoffTeams=getWinningTeams(teams,1984)  
summary(playoffTeams$r)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 619.0 729.8 783.0 789.3 847.0 1009.0

hist(playoffTeams$r,xlab="Runs Scored",main="Runs scored by playoff teams")



# Get winning batters from 1985-2015

winningBatters=getWinningBatters(teams,woba,batting,fieldingClean,salary,usatodaySalary,player,allstar,award,1984,100)

# 1) Show summary statistics and boxplots for winningBatters dataframe (ie. players from playoff teams from 1985-2015 that have batted over 100 times)

# 2) Show correlation plot of winningBatters dataframe (ie. players from playoff teams from 1985-2015 that have batted over 100 times)

showSummaryStatistics(winningBatters,names(winningBatters))

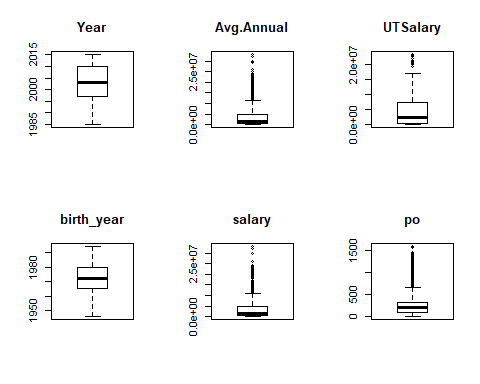
## [1] "Year"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1985 1997 2003 2002 2010 2015

## [1] "Avg.Annual"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 60000 464400 1421000 3460000 4789000 33000000

## [1] "UTSalary"  
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 0 521000 2350000 4805000 7296000 23120000 2175

## [1] "birth\_year"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1946 1965 1972 1972 1980 1994

## [1] "salary"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 60000 462500 1388000 3423000 4690000 33000000



## [1] "po"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 96.0 205.0 292.3 329.0 1597.0

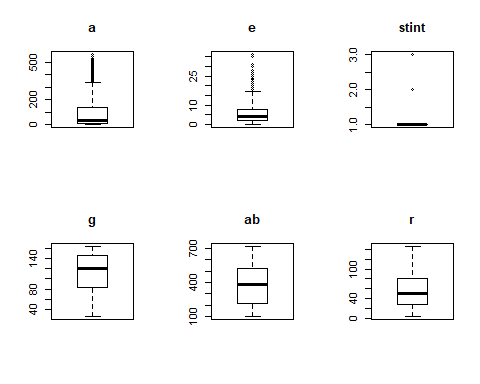
## [1] "a"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 6.00 34.00 97.64 139.00 561.00

## [1] "e"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 2.000 4.000 5.876 8.000 36.000

## [1] "stint"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 1.000 1.063 1.000 3.000

## [1] "g"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 26.0 84.0 120.5 112.8 145.0 163.0

## [1] "ab"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 101 212 385 373 526 716



## [1] "r"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.00 27.00 51.00 54.87 80.00 146.00

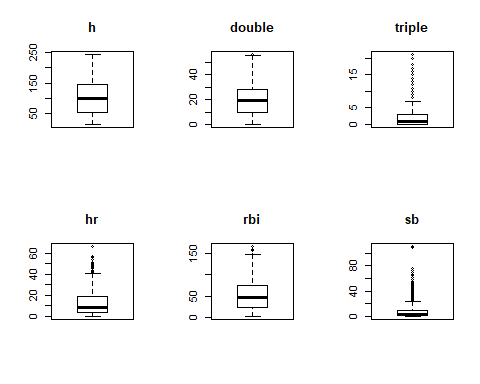
## [1] "h"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 15.0 54.0 101.0 102.2 146.0 242.0

## [1] "double"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 10.00 19.00 20.03 28.00 56.00

## [1] "triple"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 0.000 1.000 2.174 3.000 21.000

## [1] "hr"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 4.00 9.00 12.44 19.00 66.00

## [1] "rbi"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.00 25.00 47.00 52.38 74.00 165.00



## [1] "sb"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.000 3.000 7.497 10.000 110.000

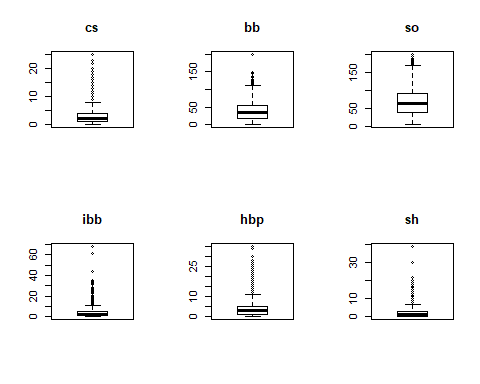
## [1] "cs"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.000 2.000 2.988 4.000 25.000

## [1] "bb"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 19.00 34.00 39.54 56.00 198.00

## [1] "so"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5.00 38.00 63.00 67.31 91.00 199.00

## [1] "ibb"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.000 2.000 3.409 5.000 68.000

## [1] "hbp"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.000 3.000 3.691 5.000 35.000



## [1] "sh"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 0.00 1.00 2.18 3.00 39.00

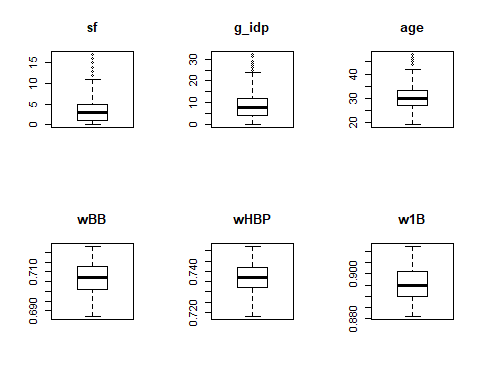
## [1] "sf"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.000 3.000 3.422 5.000 17.000

## [1] "g\_idp"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 4.000 8.000 8.671 12.000 32.000

## [1] "age"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 19.00 27.00 30.00 30.16 33.00 48.00

## [1] "wBB"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.6870 0.7010 0.7070 0.7055 0.7130 0.7230

## [1] "wHBP"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.7180 0.7320 0.7370 0.7361 0.7420 0.7520



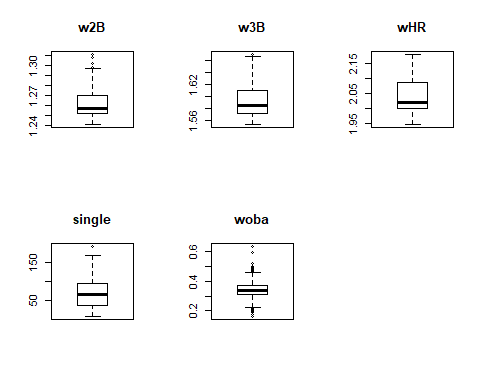
## [1] "w1B"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.8810 0.8900 0.8950 0.8953 0.9010 0.9120

## [1] "w2B"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.241 1.252 1.257 1.262 1.270 1.311

## [1] "w3B"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.554 1.572 1.585 1.593 1.611 1.670

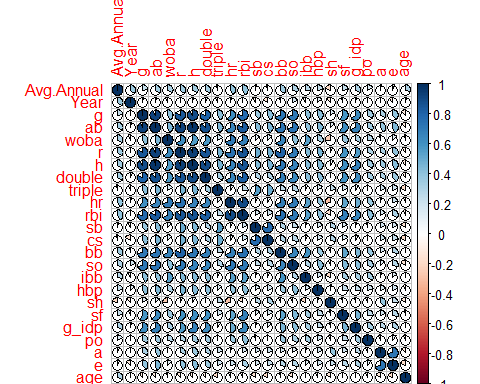
## [1] "wHR"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.947 1.999 2.021 2.039 2.086 2.178

## [1] "single"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 8.00 36.00 66.00 67.57 94.00 192.00



## [1] "woba"  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1615 0.3074 0.3366 0.3381 0.3675 0.6314

features=c("Avg.Annual","Year","g","ab","woba","r", "h", "double", "triple", "hr", "rbi", "sb","cs","bb","so","ibb","hbp", "sh","sf","g\_idp","po","a","e","age")  
battersCor=showBattersCorPlot(winningBatters,features)



# 1) Clustering will be performed with players rom playoff teams from 1985-2015 that have batted over 100 times)

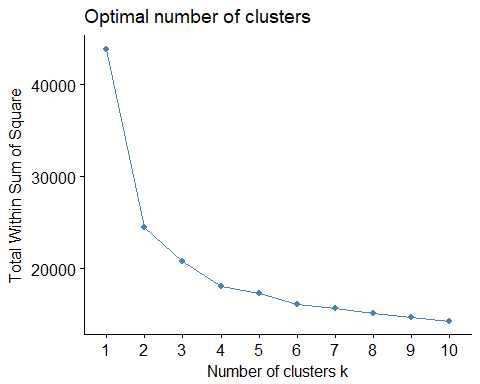
# 2) Define features for clustering, initialize data for clustering algorithms

# Features for clustering (PCA)  
featuresForClustering=c("player\_id","Year","g","ab","r", "h", "double", "triple", "hr", "rbi", "sb", "cs", "bb", "so", "ibb", "hbp", "sf","woba")  
  
# Get players by position for Kmeans and dbscan clustering, data will scaled/normalized before performing kmeans  
posAllClustering=getScaledClusterData(winningBatters,"ALL",featuresForClustering)  
posCClustering=getScaledClusterData(winningBatters,"C",featuresForClustering)  
pos1BClustering=getScaledClusterData(winningBatters,"1B",featuresForClustering)  
pos2BClustering=getScaledClusterData(winningBatters,"2B",featuresForClustering)  
posSSClustering=getScaledClusterData(winningBatters,"SS",featuresForClustering)  
pos3BClustering=getScaledClusterData(winningBatters,"3B",featuresForClustering)  
posOFClustering=getScaledClusterData(winningBatters,"OF",featuresForClustering)

# Kmeans Clustering for position of "All"

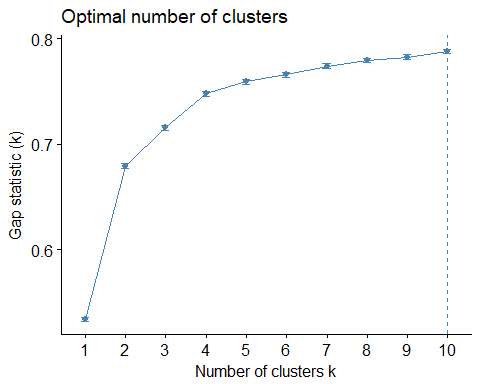
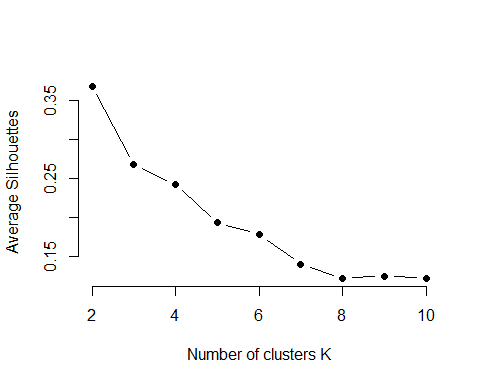
# Show Optimal clusters  
showOptimalClusters(posAllClustering)

## Warning: did not converge in 10 iterations



## NULL

## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations



# Based on clustering selection methods above and optimal clusters is "3"  
clusAll=kmeans(posAllClustering, centers=5,nstart=25)

# Dbscan Clustering for position of "All"

dbClusAll=dbscan(posAllClustering[,c(-1,-2)],eps=2,minPts = 3)

# Kmeans clustering produces 4 clusters of "All"

# \*\*\* Cluster 1: Top performers

# \*\*\* Cluster 2: Above average with speed

# \*\*\* Cluster 3: Backup or part time players

# \*\*\* Cluster 4: Above average with power

# \*\*\* Cluster 5: Average player

# Density based clustering

# \*\*\* Exceptional players are treated as noise

# From both kmeans and density based clustering, most of the players are backup are part time players

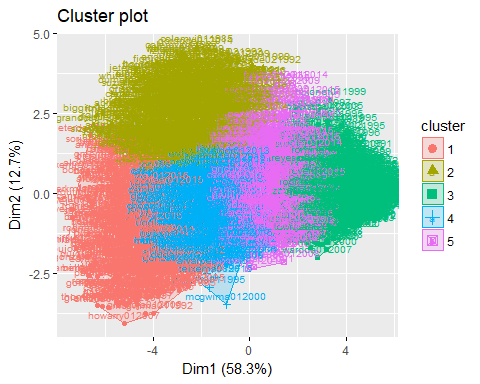
# Unscale cluster centroids  
CentersAllTransformed <- t(apply(clusAll$centers, 1, function(r) r \* attr(posAllClustering, 'scaled:scale') + attr(posAllClustering, 'scaled:center')))  
CentersAllTransformed

## g ab r h double triple hr  
## 1 150.64541 556.2577 98.26786 164.88265 34.076531 2.4604592 30.094388  
## 2 143.35492 539.3094 85.54916 151.70264 27.354916 4.8429257 12.527578  
## 3 67.39477 166.4981 20.32752 41.18929 7.775841 0.8443337 3.721046  
## 4 137.44915 488.0297 68.37500 133.54873 27.519068 1.8665254 17.720339  
## 5 108.43333 327.9061 42.07121 85.59242 16.621212 1.7227273 8.381818  
## rbi sb cs bb so ibb hbp  
## 1 105.84056 7.213010 3.047194 77.21556 103.34566 7.7959184 5.604592  
## 2 61.85851 19.273381 6.869305 52.44484 86.89928 2.5611511 4.604317  
## 3 19.04483 2.367995 1.102117 14.82565 31.93773 0.9688667 1.465753  
## 4 72.43856 4.358051 2.223517 49.11864 89.23199 4.0550847 4.811441  
## 5 40.77576 4.557576 2.309848 31.40303 60.54394 2.4257576 2.860606  
## sf woba  
## 1 6.418367 0.4014014  
## 2 4.067146 0.3420449  
## 3 1.318804 0.3061540  
## 4 4.752119 0.3499526  
## 5 2.765152 0.3273828

# Visualize kmeans cluster  
clusAll$size

## [1] 392 417 803 472 660

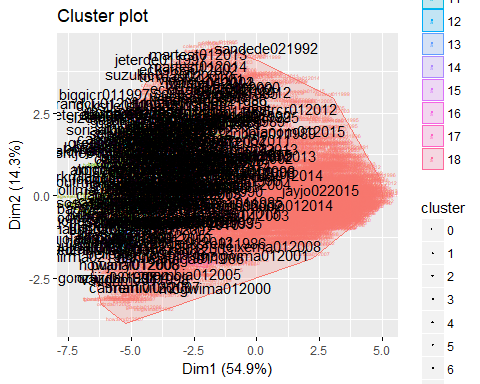
fviz\_cluster(clusAll,posAllClustering,labelsize=8)



# Visualize dbscan cluster  
dbClusAll

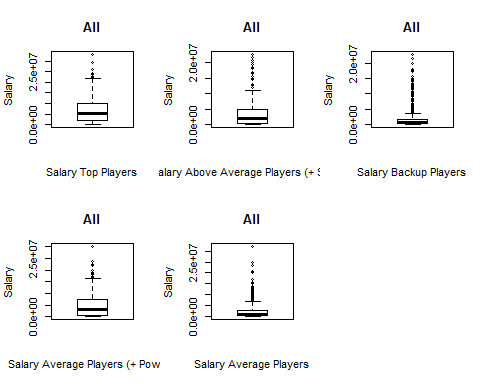
## DBSCAN clustering for 2744 objects.  
## Parameters: eps = 2, minPts = 3  
## The clustering contains 18 cluster(s) and 253 noise points.  
##   
## 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14   
## 253 2411 11 3 9 10 4 3 4 3 3 3 6 3 3   
## 15 16 17 18   
## 4 3 4 4   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClusAll,posAllClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 2)



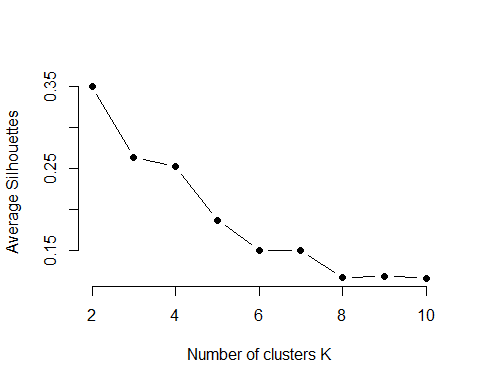
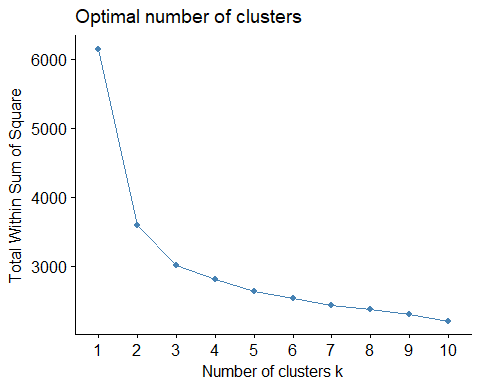
# Boxplot of player salaries per cluster for pos "ALL"

par(mfrow = c(2,3))   
boxplot(winningBatters[which(winningBatters$pos!="P"),][which(clusAll$cluster==1),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="All")  
boxplot(winningBatters[which(winningBatters$pos!="P"),][which(clusAll$cluster==2),c("Avg.Annual")],xlab="Salary Above Average Players (+ Speed)",ylab="Salary",main="All")  
boxplot(winningBatters[which(winningBatters$pos!="P"),][which(clusAll$cluster==3),c("Avg.Annual")],xlab="Salary Backup Players",ylab="Salary",main="All")  
boxplot(winningBatters[which(winningBatters$pos!="P"),][which(clusAll$cluster==4),c("Avg.Annual")],xlab="Salary Average Players (+ Power)",ylab="Salary",main="All")  
boxplot(winningBatters[which(winningBatters$pos!="P"),][which(clusAll$cluster==5),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="All")  
par(mfrow = c(1,1))

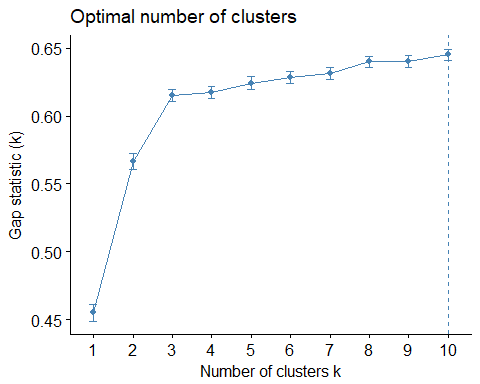


# Kmeans Clustering for position of "C"

# Show Optimal clusters  
showOptimalClusters(posCClustering)



## NULL



# Based on clustering selection methods above and optimal clusters is "3"  
clusC=kmeans(posCClustering, centers=3,nstart=25)

# Dbscan Clustering for position of "C"

dbClusC=dbscan(posCClustering[,c(-1,-2)],eps=2.15,minPts = 3.5)

# Kmeans clustering produces 3 clusters of "C"

# \*\*\* Cluster 1: Backup or part time players

# \*\*\* Cluster 2: Average players

# \*\*\* Cluster 3: Top performers

# Density based clustering

# \*\*\* Exceptional players are treated as noise

# From both kmeans and density based clustering, most of the players are backup are part time players

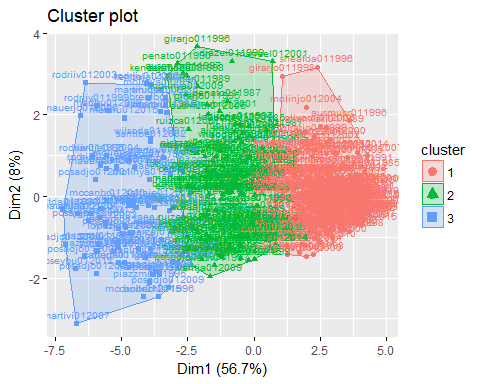
# Unscale cluster centroids  
CentersCTransformed <- t(apply(clusC$centers, 1, function(r) r \* attr(posCClustering, 'scaled:scale') + attr(posCClustering, 'scaled:center')))  
CentersCTransformed

## g ab r h double triple hr  
## 1 61.63006 166.9191 17.50289 40.54913 8.040462 0.4624277 3.895954  
## 2 112.47794 361.3382 39.63971 93.15441 18.411765 0.8639706 8.992647  
## 3 135.93421 480.4211 69.02632 138.23684 28.039474 1.0263158 19.401316  
## rbi sb cs bb so ibb hbp  
## 1 19.86127 0.5722543 0.6531792 14.08671 33.99422 1.161850 1.713873  
## 2 45.57353 1.4558824 1.3602941 33.30882 61.86765 3.352941 3.669118  
## 3 79.73684 2.4868421 1.8684211 51.78947 86.67105 5.736842 4.763158  
## sf woba  
## 1 1.433526 0.3019587  
## 2 2.937500 0.3224792  
## 3 4.394737 0.3705732

# Visualize kmeans cluster  
clusC$size

## [1] 173 136 76

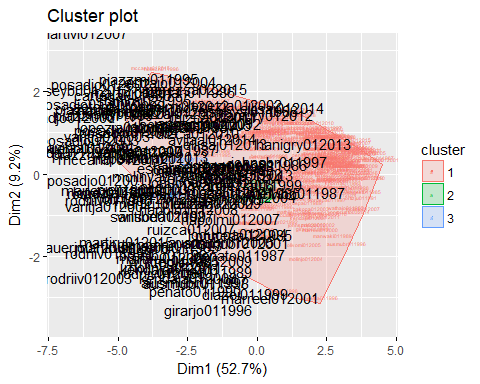
fviz\_cluster(clusC,posCClustering,labelsize=8)



# Visualize dbscan cluster  
dbClusC

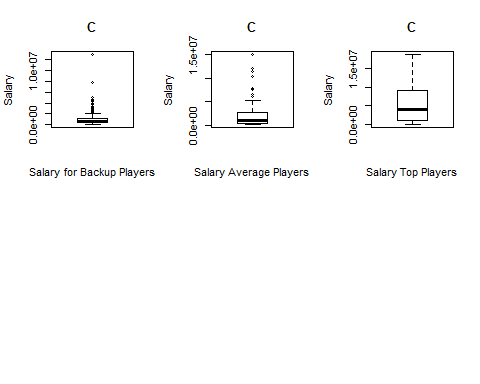
## DBSCAN clustering for 385 objects.  
## Parameters: eps = 2.15, minPts = 3.5  
## The clustering contains 3 cluster(s) and 114 noise points.  
##   
## 0 1 2 3   
## 114 263 5 3   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClusC,posCClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 1)



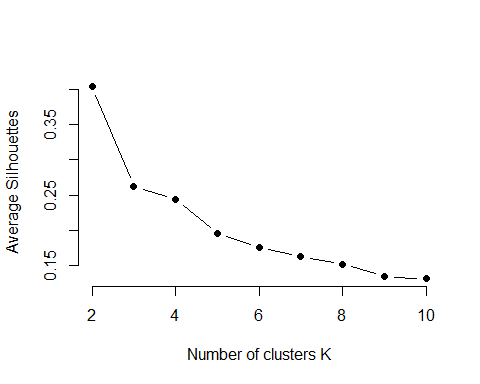
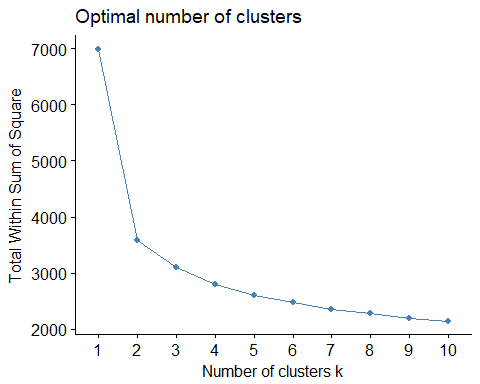
# Boxplot of player salaries per cluster for pos "C"

par(mfrow = c(2,3))  
boxplot(winningBatters[which(winningBatters$pos=="C"),][which(clusC$cluster==1),c("Avg.Annual")],xlab="Salary for Backup Players",ylab="Salary",main="C")  
boxplot(winningBatters[which(winningBatters$pos=="C"),][which(clusC$cluster==2),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="C")  
boxplot(winningBatters[which(winningBatters$pos=="C"),][which(clusC$cluster==3),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="C")  
par(mfrow = c(1,1))

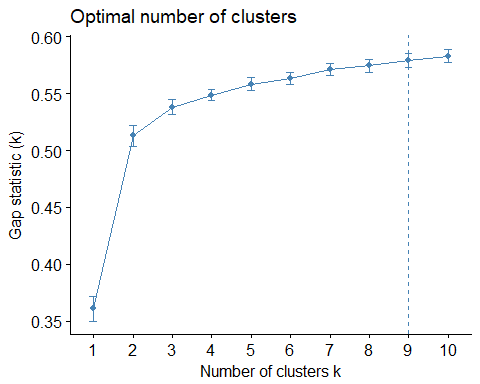


# Kmeans Clustering for position of "1B"

showOptimalClusters(pos1BClustering)



## NULL



clus1B=kmeans(pos1BClustering, centers=4,nstart=25)

# Dbscan Clustering for position of "1B"

dbClus1B=dbscan(pos1BClustering[,c(-1,-2)],eps=2,minPts = 3)

# Kmeans clustering produces 4 clusters of "1B"

# \*\*\* Cluster 1: Backup/part time or underachieving players

# \*\*\* Cluster 2: Top players

# \*\*\* Cluster 3: Average players

# \*\*\* Cluster 4: Above average players

# Density based clustering

# \*\*\* Exceptional players are treated as noise

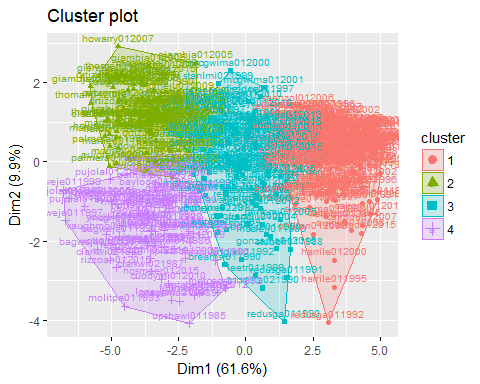
Centers1BTransformed <- t(apply(clus1B$centers, 1, function(r) r \* attr(pos1BClustering, 'scaled:scale') + attr(pos1BClustering, 'scaled:center')))  
Centers1BTransformed

## g ab r h double triple hr  
## 1 76.56471 181.5000 21.97647 47.07647 9.235294 0.7058824 5.523529  
## 2 149.97917 542.3750 90.07292 156.13542 32.645833 1.0208333 30.947917  
## 3 125.66667 400.6296 53.86111 107.42593 22.333333 1.1296296 15.222222  
## 4 151.09375 552.2969 90.95312 160.57812 32.906250 2.9687500 26.296875  
## rbi sb cs bb so ibb hbp  
## 1 24.61765 1.227941 0.6588235 18.78824 36.47647 1.370588 1.629412  
## 2 105.50000 1.854167 1.0416667 84.48958 109.13021 9.781250 5.635417  
## 3 61.46296 2.350694 1.3981481 44.75000 76.99074 3.500000 3.842593  
## 4 97.73438 6.218750 3.4531250 71.95898 101.43750 7.984375 5.609375  
## sf woba  
## 1 1.547059 0.3284993  
## 2 6.619792 0.4000119  
## 3 3.972222 0.3493319  
## 4 5.218750 0.3878672

clus1B$size

## [1] 170 96 108 64

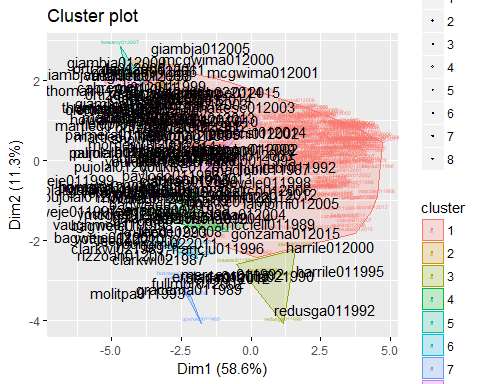
fviz\_cluster(clus1B,pos1BClustering,labelsize=8)



dbClus1B

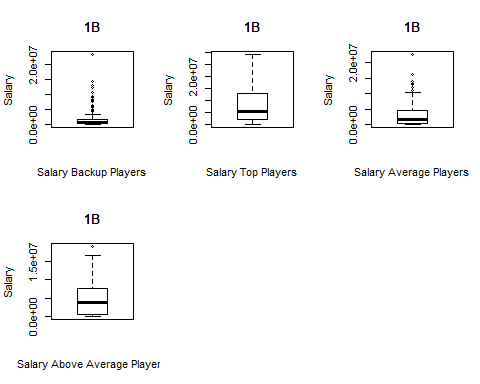
## DBSCAN clustering for 438 objects.  
## Parameters: eps = 2, minPts = 3  
## The clustering contains 9 cluster(s) and 133 noise points.  
##   
## 0 1 2 3 4 5 6 7 8 9   
## 133 274 5 5 5 3 3 4 3 3   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClus1B,pos1BClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 2)



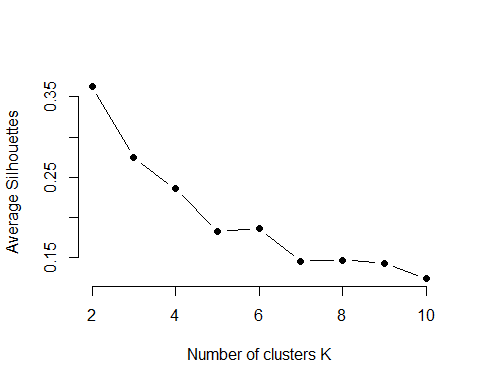
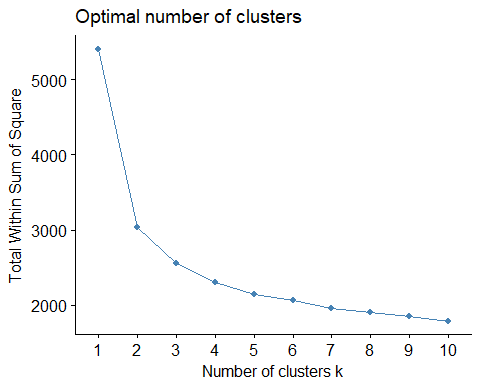
# Boxplot of player salaries per cluster for pos "1B"

par(mfrow = c(2,3))  
boxplot(winningBatters[which(winningBatters$pos=="1B"),][which(clus1B$cluster==1),c("Avg.Annual")],xlab="Salary Backup Players",ylab="Salary",main="1B")  
boxplot(winningBatters[which(winningBatters$pos=="1B"),][which(clus1B$cluster==2),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="1B")  
boxplot(winningBatters[which(winningBatters$pos=="1B"),][which(clus1B$cluster==3),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="1B")  
boxplot(winningBatters[which(winningBatters$pos=="1B"),][which(clus1B$cluster==4),c("Avg.Annual")],xlab="Salary Above Average Players",ylab="Salary",main="1B")  
par(mfrow = c(1,1))

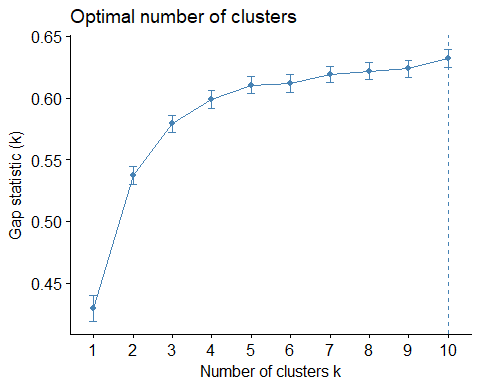


# Kmeans Clustering for position of "2B"

showOptimalClusters(pos2BClustering)



## NULL



clus2B=kmeans(pos2BClustering, centers=4,nstart=25)

# Dbscan Clustering for position of "2B"

dbClus2B=dbscan(pos2BClustering[,c(-1,-2)],eps=2,minPts = 3)

# Kmeans clustering produces 3 clusters of "2B"

# \*\*\* Cluster 1: Backup, part time, or underachieving players

# \*\*\* Cluster 2: Above average player

# \*\*\* Cluster 3: Average players

# \*\*\* Cluster 4: Top performers

# Density based clustering

# \*\*\* Exceptional players or above average players are treated as noise

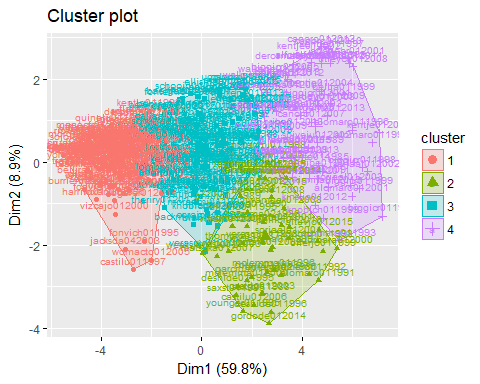
Centers2BTransformed <- t(apply(clus2B$centers, 1, function(r) r \* attr(pos2BClustering, 'scaled:scale') + attr(pos2BClustering, 'scaled:center')))  
Centers2BTransformed

## g ab r h double triple hr  
## 1 77.40984 191.4262 24.62295 47.16393 8.762295 1.245902 2.377049  
## 2 143.08163 535.4082 80.59184 149.77551 24.775510 5.673469 8.224490  
## 3 121.89908 407.0092 54.11927 109.26606 21.000000 2.252294 7.385321  
## 4 153.05085 595.1186 104.03390 174.49153 38.915254 3.635593 19.927966  
## rbi sb cs bb so ibb hbp  
## 1 18.14754 3.631148 1.754098 16.12295 33.40164 0.7704918 1.573770  
## 2 52.26531 20.071429 8.204082 52.12245 75.40816 2.5612245 5.724490  
## 3 44.82569 6.669725 3.000000 36.91743 63.31881 2.3027523 4.261468  
## 4 86.66102 14.033898 4.542373 61.67797 94.72881 3.5000000 7.644068  
## sf woba  
## 1 1.516393 0.2926713  
## 2 3.897959 0.3340776  
## 3 3.431193 0.3226799  
## 4 6.135593 0.3701497

clus2B$size

## [1] 122 49 109 59

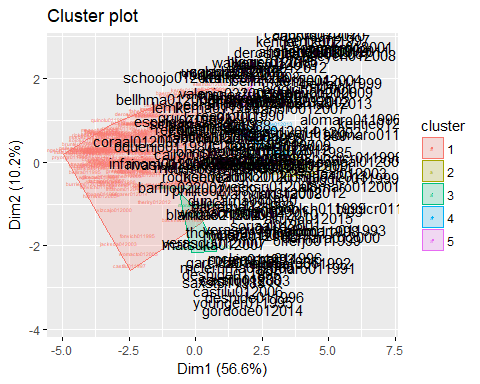
fviz\_cluster(clus2B,pos2BClustering,labelsize=8)



dbClus2B

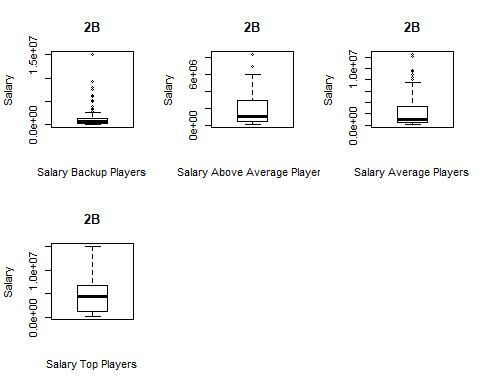
## DBSCAN clustering for 339 objects.  
## Parameters: eps = 2, minPts = 3  
## The clustering contains 5 cluster(s) and 134 noise points.  
##   
## 0 1 2 3 4 5   
## 134 191 3 4 4 3   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClus2B,pos2BClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 2)



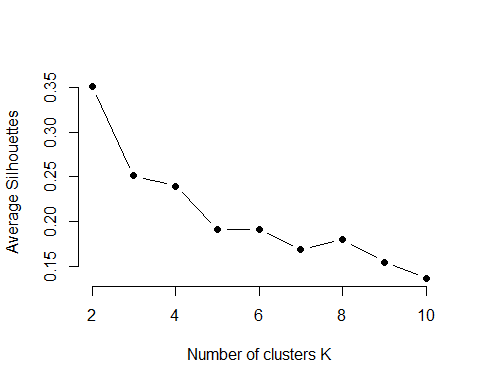
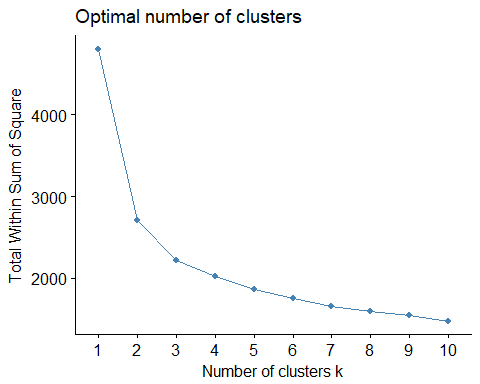
# Boxplot of player salaries per cluster for pos "2B"

par(mfrow = c(2,3))  
boxplot(winningBatters[which(winningBatters$pos=="2B"),][which(clus2B$cluster==1),c("Avg.Annual")],xlab="Salary Backup Players",ylab="Salary",main="2B")  
boxplot(winningBatters[which(winningBatters$pos=="2B"),][which(clus2B$cluster==2),c("Avg.Annual")],xlab="Salary Above Average Players",ylab="Salary",main="2B")  
boxplot(winningBatters[which(winningBatters$pos=="2B"),][which(clus2B$cluster==3),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="2B")  
boxplot(winningBatters[which(winningBatters$pos=="2B"),][which(clus2B$cluster==4),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="2B")  
par(mfrow = c(1,1))

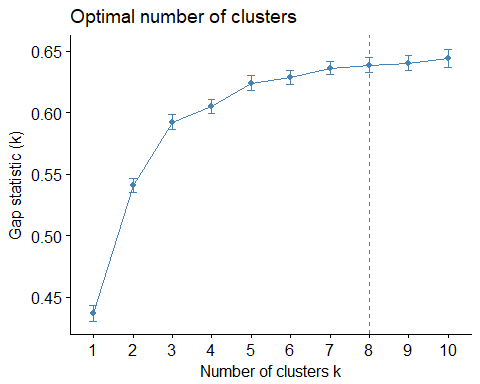


# Kmeans Clustering for position of "3B"

showOptimalClusters(pos3BClustering)



## NULL



clus3B=kmeans(pos3BClustering, centers=4,nstart=25)

# Dbscan Clustering for position of "3B"

dbClus3B=dbscan(pos3BClustering[,c(-1,-2)],eps=2,minPts = 3)

# Kmeans clustering produces 5 clusters of "3B"

# \*\*\* Cluster 1: Average players with speed

# \*\*\* Cluster 2: Backup/part time players

# \*\*\* Cluster 3: Average players

# \*\*\* Cluster 4: Top players

# Density based clustering

# \*\*\* Exceptional players are treated as noise

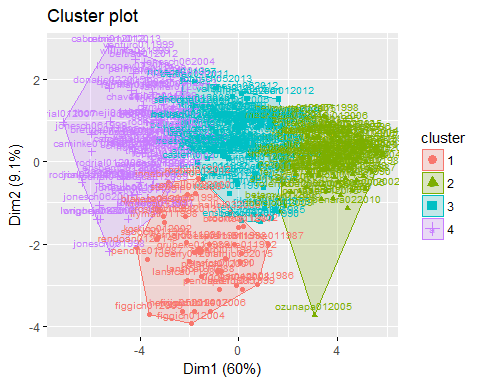
Centers3BTransformed <- t(apply(clus3B$centers, 1, function(r) r \* attr(pos3BClustering, 'scaled:scale') + attr(pos3BClustering, 'scaled:center')))  
Centers3BTransformed

## g ab r h double triple hr  
## 1 136.69231 490.0769 73.46154 135.10256 24.871795 3.1538462 12.846154  
## 2 68.78788 187.5960 22.41414 45.94949 9.060606 0.8484848 5.121212  
## 3 124.32584 417.2022 53.91011 112.71910 22.932584 1.4157303 12.932584  
## 4 149.55405 555.5811 93.58108 162.44595 33.256757 2.4324324 27.864865  
## rbi sb cs bb so ibb hbp sf  
## 1 61.69231 10.435897 5.076923 47.00000 82.66667 2.230769 3.923077 3.538462  
## 2 23.75758 1.540404 1.020202 15.83838 36.01010 1.101010 1.525253 1.424242  
## 3 58.44944 2.134831 1.505618 39.47191 71.04494 2.662921 3.247191 4.606742  
## 4 99.75676 5.858108 2.756757 70.89189 99.62838 7.135135 4.270270 6.283784  
## woba  
## 1 0.3393336  
## 2 0.3057489  
## 3 0.3377466  
## 4 0.3881225

clus3B$size

## [1] 39 99 89 74

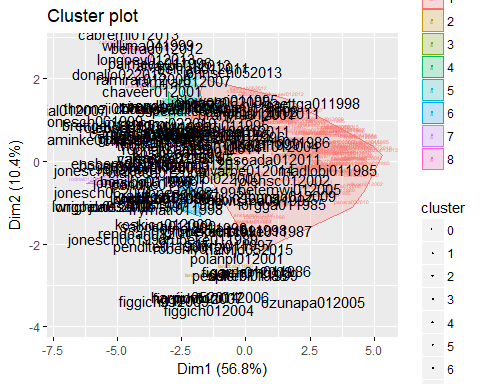
fviz\_cluster(clus3B,pos3BClustering,labelsize=8)



dbClus3B

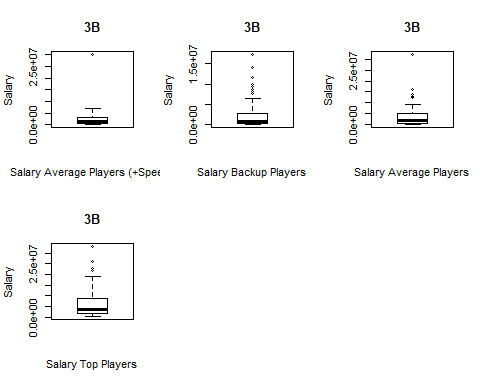
## DBSCAN clustering for 301 objects.  
## Parameters: eps = 2, minPts = 3  
## The clustering contains 8 cluster(s) and 111 noise points.  
##   
## 0 1 2 3 4 5 6 7 8   
## 111 164 3 3 4 6 4 3 3   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClus3B,pos3BClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 2)



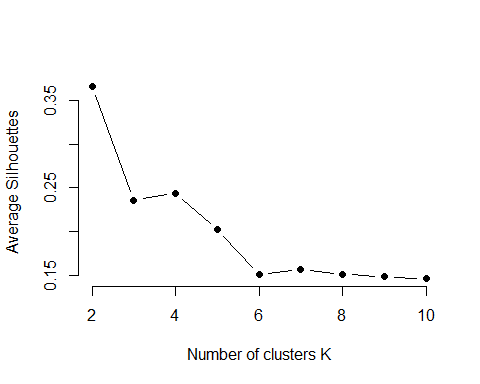
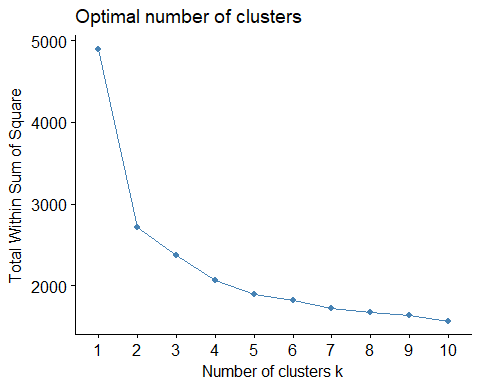
# Boxplot of player salaries per cluster for pos "3B"

par(mfrow = c(2,3))  
boxplot(winningBatters[which(winningBatters$pos=="3B"),][which(clus3B$cluster==1),c("Avg.Annual")],xlab="Salary Average Players (+Speed)",ylab="Salary",main="3B")  
boxplot(winningBatters[which(winningBatters$pos=="3B"),][which(clus3B$cluster==2),c("Avg.Annual")],xlab="Salary Backup Players",ylab="Salary",main="3B")  
boxplot(winningBatters[which(winningBatters$pos=="3B"),][which(clus3B$cluster==3),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="3B")  
boxplot(winningBatters[which(winningBatters$pos=="3B"),][which(clus3B$cluster==4),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="3B")  
par(mfrow = c(1,1))

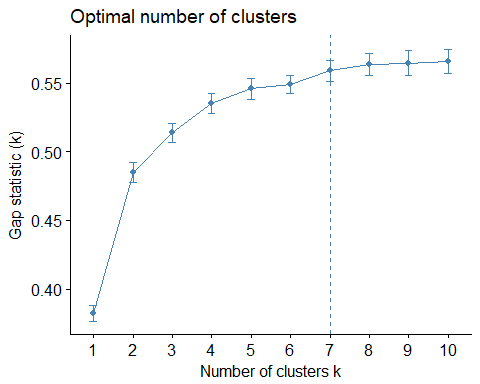


# Kmeans Clustering for position of "SS"

showOptimalClusters(posSSClustering)



## NULL



clusSS=kmeans(posSSClustering, centers=4,nstart=25)

# Dbscan Clustering for position of "SS"

dbClusSS=dbscan(posSSClustering[,c(-1,-2)],eps=2,minPts = 3)

# Kmeans clustering produces 3 clusters of "SS"

# \*\*\* Cluster 1: Average players

# \*\*\* Cluster 2: Above average players

# \*\*\* Cluster 3: Top performers

# \*\*\* Cluster 4: Backup or part time players

# Density based clustering

# \*\*\* Exceptional players or above average are treated as noise

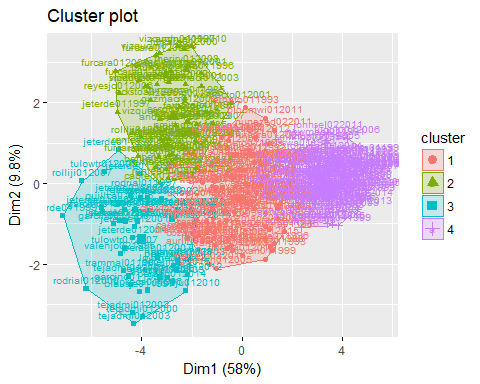
CentersSSTransformed <- t(apply(clusSS$centers, 1, function(r) r \* attr(posSSClustering, 'scaled:scale') + attr(posSSClustering, 'scaled:center')))  
CentersSSTransformed

## g ab r h double triple hr  
## 1 125.62637 424.1429 53.37363 109.47253 21.16484 2.494505 7.142857  
## 2 149.03279 578.3770 86.91803 160.91803 27.65574 5.180328 7.918033  
## 3 151.33333 589.4375 97.87500 174.79167 33.08333 3.208333 20.854167  
## 4 73.60748 196.0561 23.85047 47.70093 8.35514 1.149533 2.663551  
## rbi sb cs bb so ibb hbp sf  
## 1 44.29670 6.703297 3.252747 36.79121 70.10989 2.615385 3.681319 3.604396  
## 2 57.59016 24.381148 8.606557 50.50820 74.26230 1.688525 3.393443 4.500000  
## 3 85.60938 14.296875 4.145833 56.18750 98.37500 3.343750 6.854167 4.385417  
## 4 18.60748 3.308411 1.579439 16.52336 31.19626 1.084112 1.485981 1.373832  
## woba  
## 1 0.3114815  
## 2 0.3233082  
## 3 0.3656382  
## 4 0.2896336

clusSS$size

## [1] 91 61 48 107

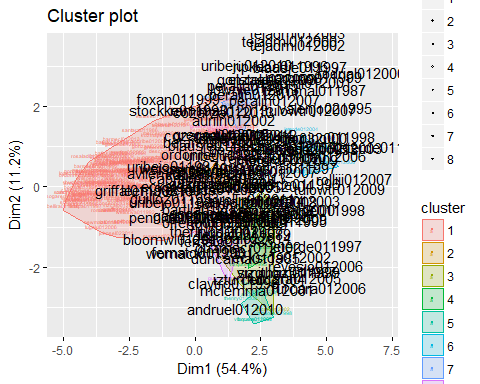
fviz\_cluster(clusSS,posSSClustering,labelsize=8)



dbClusSS

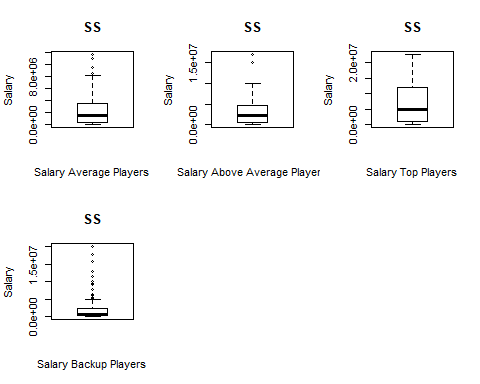
## DBSCAN clustering for 307 objects.  
## Parameters: eps = 2, minPts = 3  
## The clustering contains 9 cluster(s) and 110 noise points.  
##   
## 0 1 2 3 4 5 6 7 8 9   
## 110 158 3 9 4 7 6 4 3 3   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClusSS,posSSClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 2)



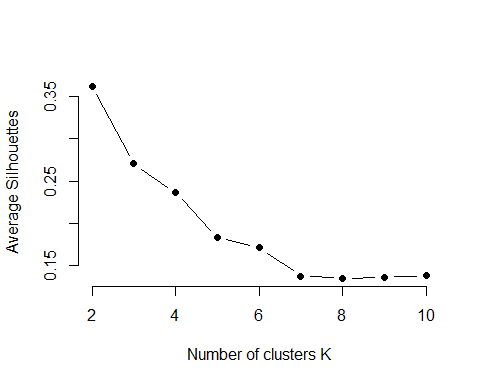
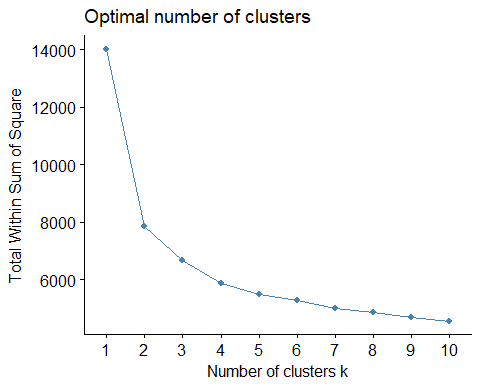
# Boxplot of player salaries per cluster for pos "SS"

par(mfrow = c(2,3))  
boxplot(winningBatters[which(winningBatters$pos=="SS"),][which(clusSS$cluster==1),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="SS")  
boxplot(winningBatters[which(winningBatters$pos=="SS"),][which(clusSS$cluster==2),c("Avg.Annual")],xlab="Salary Above Average Players",ylab="Salary",main="SS")  
boxplot(winningBatters[which(winningBatters$pos=="SS"),][which(clusSS$cluster==3),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="SS")  
boxplot(winningBatters[which(winningBatters$pos=="SS"),][which(clusSS$cluster==4),c("Avg.Annual")],xlab="Salary Backup Players",ylab="Salary",main="SS")  
par(mfrow = c(1,1))

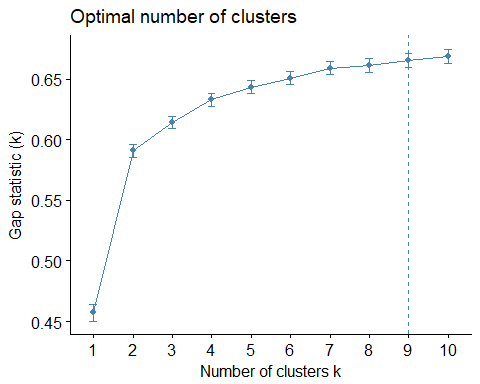


# Kmeans Clustering for position of "OF"

showOptimalClusters(posOFClustering)



## NULL



clusOF=kmeans(posOFClustering, centers=5,nstart=25)

# Dbscan Clustering for position of "OF"

dbClusOF=dbscan(posOFClustering[,c(-1,-2)],eps=1.95,minPts = 3)

# Kmeans clustering produces 5 clusters of "OF"

# \*\*\* Cluster 1: Average player

# \*\*\* Cluster 2: Backup, part time, or underachievers players

# \*\*\* Cluster 3: Top performers

# \*\*\* Cluster 4: Average player (with speed)

# \*\*\* Cluster 5: Above average players

# Density based clustering

# \*\*\* Exceptional players, above average, and speedy players are treated as noise

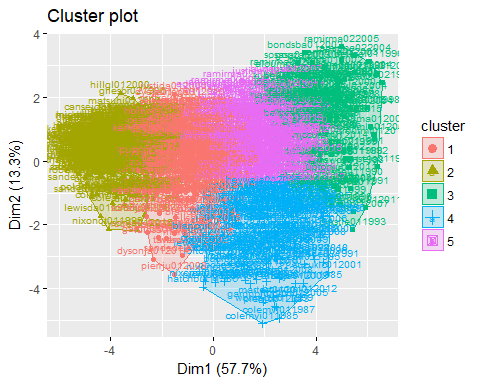
CentersOFTransformed <- t(apply(clusOF$centers, 1, function(r) r \* attr(posOFClustering, 'scaled:scale') + attr(posOFClustering, 'scaled:center')))  
CentersOFTransformed

## g ab r h double triple hr  
## 1 107.62621 327.9563 46.79126 87.84466 16.79126 2.553398 9.660194  
## 2 73.42797 170.7119 23.16102 42.77966 8.15678 1.135593 4.165254  
## 3 150.60800 555.7440 100.81600 164.88800 34.13600 3.372000 31.232000  
## 4 143.76119 536.7090 86.94030 150.20896 26.96269 5.880597 11.895522  
## 5 139.22727 500.0909 75.83523 137.60227 27.56818 2.406250 20.732955  
## rbi sb cs bb so ibb hbp  
## 1 42.62136 8.131068 3.160194 31.81553 65.93689 2.1019417 2.859223  
## 2 19.66525 4.322034 1.711864 16.10169 34.58475 0.8474576 1.512712  
## 3 106.45600 11.832000 4.536000 77.51200 107.74000 7.9920000 5.712000  
## 4 59.29104 27.089552 8.697761 52.77239 95.55970 2.1194030 4.343284  
## 5 77.13636 8.022727 3.619318 53.05398 96.26705 4.2840909 4.414773  
## sf woba  
## 1 2.703883 0.3378778  
## 2 1.288136 0.3139513  
## 3 6.440000 0.4041912  
## 4 3.686567 0.3411165  
## 5 4.897727 0.3579107

clusOF$size

## [1] 206 236 125 134 176

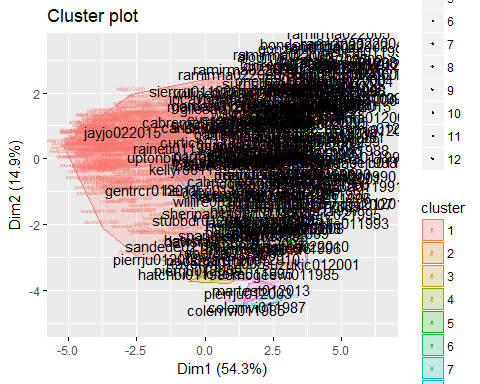
fviz\_cluster(clusOF,posOFClustering,labelsize=8)



dbClusOF

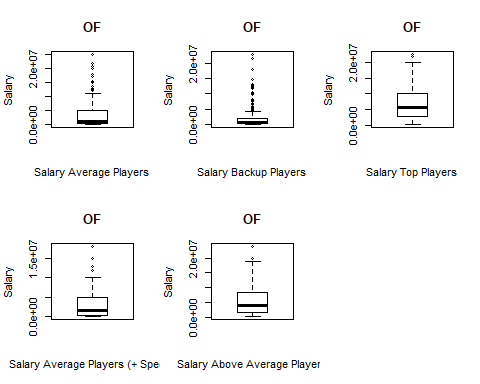
## DBSCAN clustering for 877 objects.  
## Parameters: eps = 1.95, minPts = 3  
## The clustering contains 13 cluster(s) and 219 noise points.  
##   
## 0 1 2 3 4 5 6 7 8 9 10 11 12 13   
## 219 609 4 5 4 9 3 3 3 5 3 3 4 3   
##   
## Available fields: cluster, eps, minPts

fviz\_cluster(dbClusOF,posOFClustering[,c(-1,-2)],axes=c(1,2),geom="text",labelsize = 4,outlier.shape = 2)



# Boxplot of player salaries per cluster for pos "OF"

par(mfrow = c(2,3))  
boxplot(winningBatters[which(winningBatters$pos=="OF"),][which(clusOF$cluster==1),c("Avg.Annual")],xlab="Salary Average Players",ylab="Salary",main="OF")  
boxplot(winningBatters[which(winningBatters$pos=="OF"),][which(clusOF$cluster==2),c("Avg.Annual")],xlab="Salary Backup Players",ylab="Salary",main="OF")  
boxplot(winningBatters[which(winningBatters$pos=="OF"),][which(clusOF$cluster==3),c("Avg.Annual")],xlab="Salary Top Players",ylab="Salary",main="OF")  
boxplot(winningBatters[which(winningBatters$pos=="OF"),][which(clusOF$cluster==4),c("Avg.Annual")],xlab="Salary Average Players (+ Speed)",ylab="Salary",main="OF")  
boxplot(winningBatters[which(winningBatters$pos=="OF"),][which(clusOF$cluster==5),c("Avg.Annual")],xlab="Salary Above Average Players",ylab="Salary",main="OF")  
par(mfrow = c(1,1))



# Setup cross folds verification for linear and random forest regression models

# sETUP 10-fold crossvalidationf or lm models  
modelCtrlLM <- trainControl(method='cv', number=10,summaryFunction=modelSummary)  
  
# SETUP 5 fold crossvalidation for rf models (5 fold because rf takes a long time to run)  
modelCtrlRF <- trainControl(method='cv', number=5,summaryFunction=modelSummary)

# Basic Model with most correlated attributes to Avg.Annual

# RMSE Rsquared ME MAE MPE MAPE

# 3437541 0.4473659 -1158.883 2423509 -8.005655 321.2389

# > summary(basicModelError$pred.err.percent.basicModel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.066 35.720 95.430 319.500 357.600 7874.000

basicModelData=winningBatters[which(winningBatters$pos!="P"),names(which(abs(battersCor[1,]) > .2))]  
basicModel=train(basicModelData[,-1],basicModelData[,"Avg.Annual"],method='lm',trControl=modelCtrlLM)  
   
summary(basicModel$finalModel)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -10151470 -2139452 -479970 1415646 24359559   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -406799455 17528629 -23.208 < 2e-16 \*\*\*  
## Year 198229 8681 22.835 < 2e-16 \*\*\*  
## g -61671 5077 -12.147 < 2e-16 \*\*\*  
## ab 17765 3400 5.224 1.88e-07 \*\*\*  
## woba 6248619 3308766 1.889 0.05906 .   
## r 23997 8868 2.706 0.00685 \*\*   
## h -17746 10975 -1.617 0.10601   
## double -38102 14224 -2.679 0.00743 \*\*   
## hr 17981 20568 0.874 0.38206   
## rbi 17421 8823 1.975 0.04842 \*   
## bb 15422 5880 2.623 0.00877 \*\*   
## so -4692 3748 -1.252 0.21071   
## ibb 96520 19860 4.860 1.24e-06 \*\*\*  
## hbp 39150 19258 2.033 0.04215 \*   
## sh -76332 25437 -3.001 0.00272 \*\*   
## sf -36788 35999 -1.022 0.30692   
## g\_idp 76732 17678 4.341 1.47e-05 \*\*\*  
## age 348766 17090 20.408 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3431000 on 2726 degrees of freedom  
## Multiple R-squared: 0.4524, Adjusted R-squared: 0.4489   
## F-statistic: 132.5 on 17 and 2726 DF, p-value: < 2.2e-16

basicModel

## Linear Regression   
##   
## 2744 samples  
## 17 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2468, 2470, 2470, 2470, 2471, 2469, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 3435813 0.4470074 -4830.419 2426813 -8.749043 321.7537  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

basicModelError=getModelPredictionError(basicModel,"Avg.Annual","basicModel",basicModelData)  
summary(basicModelError$pred.err.percent.basicModel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.066 35.720 95.430 319.500 357.600 7874.000

# Removed outliers from base model

# RMSE Rsquared ME MAE MPE MAPE

# 1992829 0.4294602 -777.6134 1529543 -58.7159 207.4768

# Tuning parameter 'intercept' was held constant at a value of TRUE

# > summary(remove.outliers.basicModelError$pred.err.percent.basicModel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.069 34.460 72.260 205.600 263.200 3989.000

remove.outliers.basicModelData=removeOutliers(basicModelData,"Avg.Annual")  
remove.outliers.basicModel=train(remove.outliers.basicModelData[,-1],remove.outliers.basicModelData[,"Avg.Annual"],method='lm',trControl=modelCtrlLM)  
   
summary(remove.outliers.basicModel$finalModel)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5317389 -1287783 -307581 1019354 8477218   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -218551094 10706980 -20.412 < 2e-16 \*\*\*  
## Year 106312 5292 20.090 < 2e-16 \*\*\*  
## g -41179 3000 -13.727 < 2e-16 \*\*\*  
## ab 16186 2069 7.822 7.59e-15 \*\*\*  
## woba 3336291 1992269 1.675 0.094134 .   
## r 14351 5503 2.608 0.009161 \*\*   
## h -20483 6766 -3.027 0.002492 \*\*   
## double -9755 8726 -1.118 0.263699   
## hr 27623 12606 2.191 0.028518 \*   
## rbi 5654 5441 1.039 0.298774   
## bb 10743 3623 2.965 0.003054 \*\*   
## so -7409 2290 -3.235 0.001232 \*\*   
## ibb 31597 13821 2.286 0.022332 \*   
## hbp -3433 11875 -0.289 0.772541   
## sh -57631 15087 -3.820 0.000137 \*\*\*  
## sf 4220 22022 0.192 0.848056   
## g\_idp 8272 11082 0.746 0.455441   
## age 219797 10352 21.233 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1986000 on 2503 degrees of freedom  
## Multiple R-squared: 0.4353, Adjusted R-squared: 0.4314   
## F-statistic: 113.5 on 17 and 2503 DF, p-value: < 2.2e-16

remove.outliers.basicModel

## Linear Regression   
##   
## 2521 samples  
## 17 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2268, 2269, 2269, 2270, 2268, 2269, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 1992715 0.4281318 1027.152 1528815 -58.55913 207.4717  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

remove.outliers.basicModelError=getModelPredictionError(remove.outliers.basicModel,"Avg.Annual","basicModel",remove.outliers.basicModelData)  
summary(remove.outliers.basicModelError$pred.err.percent.basicModel)

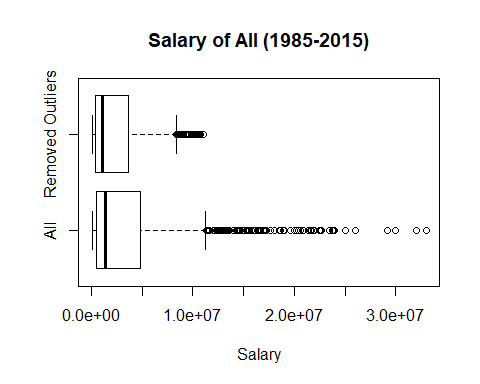
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.069 34.460 72.260 205.600 263.200 3989.000

# Setup and populate data frame for regression model analysis of all players

regressionFeatures=c("Avg.Annual","Year","woba","pos","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
  
posAllR=winningBatters[which(winningBatters$pos!="P"),]  
posAllR=posAllR[,regressionFeatures]  
posAllR$pos=factor(posAllR$pos,exclude=NA)

# Show boxplot of salaries and determine if there are outliers

posAllR.removed.outliers=removeOutliers(posAllR,"Avg.Annual")  
  
boxplot(list(posAllR$Avg.Annual,posAllR.removed.outliers$Avg.Annual),xlab="Salary",names=c("All","Removed Outliers"),main="Salary of All (1985-2015)",horizontal=TRUE)



# Normal model for All players

# RMSE Rsquared ME MAE MPE MAPE

# 3262775 0.5046987 3781.319 2313310 35.47043 317.4001

# Tuning parameter 'intercept' was held constant at a value of TRUE

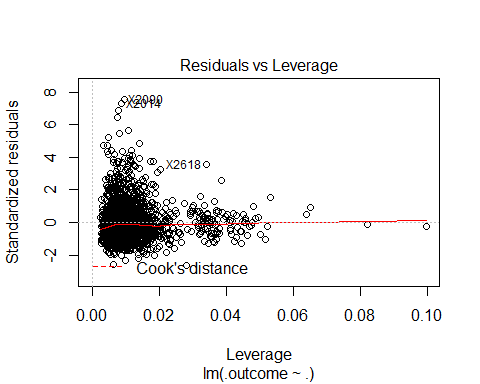
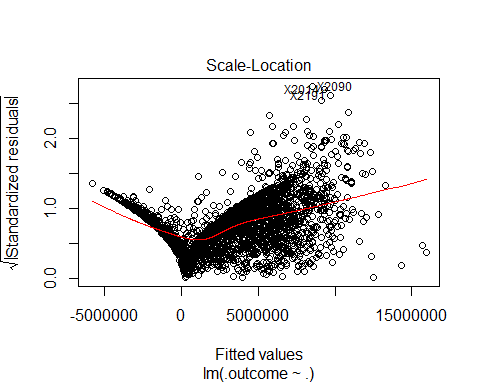
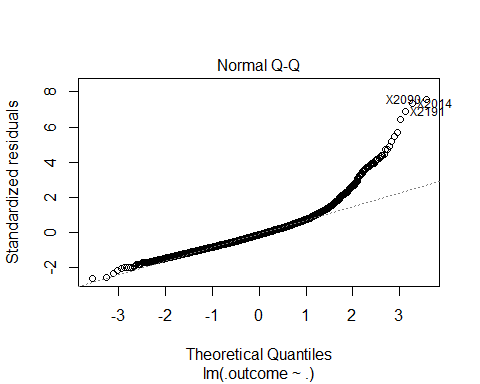
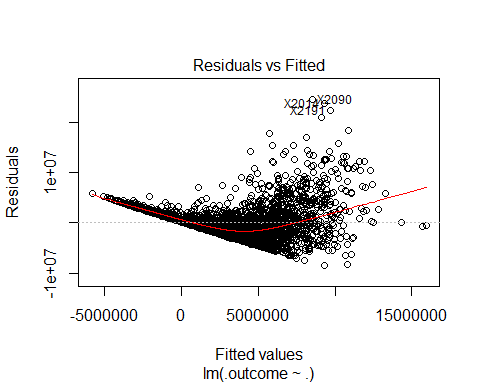
# > summary(pAllModel1Error$pred.err.percent.pAllModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.003 34.440 94.420 313.800 323.700 8208.000

full=lm(Avg.Annual~.,data=posAllR)  
null=lm(Avg.Annual~1,data=posAllR)  
pAllModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","pAllModel1",posAllR,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "isAllStar"   
## [4] "ibb" "g" "ab"   
## [7] "bb" "pos" "triple"   
## [10] "ageUnder25" "age25to30" "g\_idp"   
## [13] "hbp" "double" "woba"   
## [16] "cs" "sh" "po"   
## [19] "r" "h"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8406518 -2045885 -439159 1339303 24444337   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -3.898e+08 1.638e+07 -23.803 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 2.544e+06 1.895e+05 13.425 < 2e-16 \*\*\*  
## Year 1.937e+05 8.122e+03 23.853 < 2e-16 \*\*\*  
## isAllStarTRUE 1.231e+06 2.081e+05 5.919 3.65e-09 \*\*\*  
## ibb 8.815e+04 1.826e+04 4.829 1.45e-06 \*\*\*  
## g -5.241e+04 4.970e+03 -10.547 < 2e-16 \*\*\*  
## ab 1.827e+04 2.820e+03 6.480 1.08e-10 \*\*\*  
## bb 8.997e+03 5.381e+03 1.672 0.094638 .   
## pos2B 2.037e+04 3.037e+05 0.067 0.946542   
## pos3B 6.123e+05 3.215e+05 1.905 0.056944 .   
## posC -8.580e+05 2.418e+05 -3.548 0.000394 \*\*\*  
## posCF 4.057e+05 6.919e+05 0.586 0.557634   
## posLF -5.570e+05 6.095e+05 -0.914 0.360864   
## posOF 9.032e+05 2.542e+05 3.552 0.000388 \*\*\*  
## posRF 4.705e+05 5.897e+05 0.798 0.425051   
## posSS 1.065e+06 3.241e+05 3.287 0.001025 \*\*   
## triple -1.781e+05 3.345e+04 -5.324 1.10e-07 \*\*\*  
## ageUnder25TRUE -1.572e+06 3.096e+05 -5.078 4.07e-07 \*\*\*  
## age25to30TRUE -7.249e+05 1.863e+05 -3.892 0.000102 \*\*\*  
## g\_idp 6.161e+04 1.701e+04 3.623 0.000297 \*\*\*  
## hbp 3.827e+04 1.863e+04 2.055 0.040022 \*   
## double -4.752e+04 1.353e+04 -3.511 0.000454 \*\*\*  
## woba 1.035e+07 2.755e+06 3.758 0.000175 \*\*\*  
## cs -7.418e+04 2.539e+04 -2.922 0.003505 \*\*   
## sh -6.613e+04 2.427e+04 -2.725 0.006467 \*\*   
## po 8.264e+02 3.651e+02 2.263 0.023688 \*   
## r 2.149e+04 7.989e+03 2.690 0.007191 \*\*   
## h -1.710e+04 8.709e+03 -1.964 0.049683 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3253000 on 2716 degrees of freedom  
## Multiple R-squared: 0.5096, Adjusted R-squared: 0.5047   
## F-statistic: 104.5 on 27 and 2716 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 2744 samples  
## 20 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2471, 2469, 2469, 2469, 2470, 2470, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 3261424 0.5044456 -1598.066 2315308 34.03482 316.9175  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(pAllModel1Error$pred.err.percent.pAllModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.003 34.440 94.420 313.800 323.700 8208.000

# With salary outliers removed for all players

# RMSE Rsquared ME MAE MPE MAPE

# 1866542 0.4983523 -526.824 1432642 -24.40724 200.044

# Tuning parameter 'intercept' was held constant at a value of TRUE

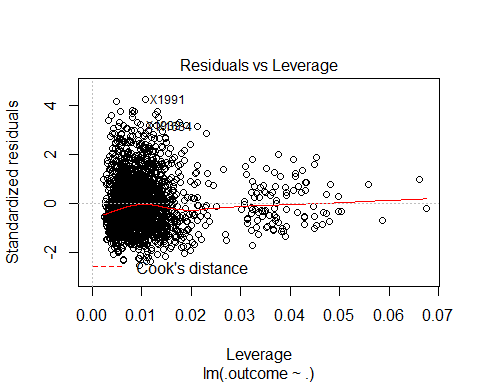
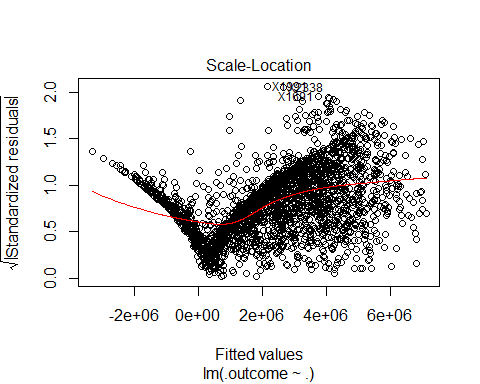
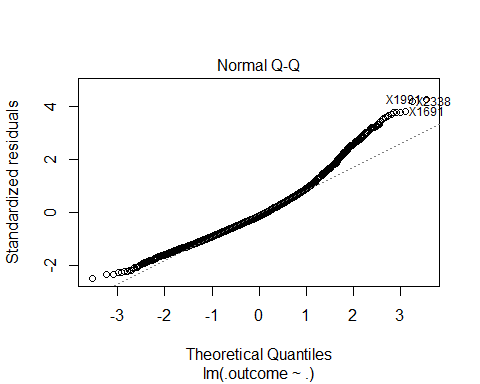
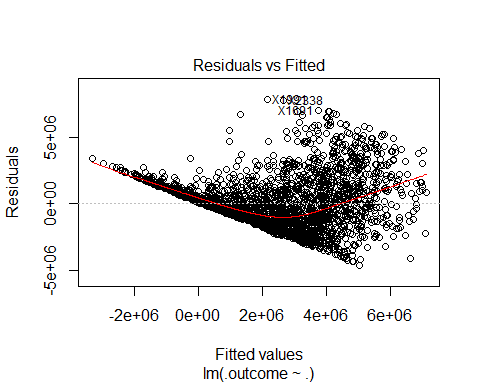
# > summary(error.model.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.034 32.550 74.400 197.900 226.500 4854.000

full=lm(Avg.Annual~.,data=posAllR.removed.outliers)  
null=lm(Avg.Annual~1,data=posAllR.removed.outliers)  
error.model.posAllR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","pAll.removed.outliers",posAllR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "isAllStar"   
## [4] "bb" "g" "ab"   
## [7] "ageUnder25" "pos" "sh"   
## [10] "isAwardWinner" "h" "age25to30"   
## [13] "woba" "so" "po"   
## [16] "triple" "hr"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4616853 -1232972 -271456 965058 7835592   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.204e+08 9.892e+06 -22.277 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 1.771e+06 1.118e+05 15.839 < 2e-16 \*\*\*  
## Year 1.097e+05 4.898e+03 22.405 < 2e-16 \*\*\*  
## isAllStarTRUE 5.890e+05 1.366e+05 4.311 1.69e-05 \*\*\*  
## bb 1.294e+04 2.795e+03 4.631 3.82e-06 \*\*\*  
## g -3.498e+04 2.847e+03 -12.289 < 2e-16 \*\*\*  
## ab 1.613e+04 1.982e+03 8.140 6.15e-16 \*\*\*  
## ageUnder25TRUE -1.132e+06 1.795e+05 -6.309 3.30e-10 \*\*\*  
## pos2B 3.460e+05 1.792e+05 1.931 0.053592 .   
## pos3B 3.034e+05 1.927e+05 1.575 0.115482   
## posC -2.959e+05 1.434e+05 -2.064 0.039078 \*   
## posCF -1.787e+04 4.029e+05 -0.044 0.964633   
## posLF -1.906e+05 3.542e+05 -0.538 0.590487   
## posOF 6.329e+05 1.500e+05 4.220 2.53e-05 \*\*\*  
## posRF 7.935e+05 3.473e+05 2.285 0.022425 \*   
## posSS 7.898e+05 1.933e+05 4.086 4.52e-05 \*\*\*  
## sh -6.511e+04 1.429e+04 -4.555 5.50e-06 \*\*\*  
## isAwardWinnerTRUE 4.378e+05 1.344e+05 3.257 0.001141 \*\*   
## h -2.171e+04 5.764e+03 -3.766 0.000170 \*\*\*  
## age25to30TRUE -3.244e+05 1.112e+05 -2.919 0.003547 \*\*   
## woba 5.208e+06 1.861e+06 2.798 0.005174 \*\*   
## so -6.970e+03 2.102e+03 -3.316 0.000926 \*\*\*  
## po 6.041e+02 2.257e+02 2.676 0.007493 \*\*   
## triple -3.854e+04 1.904e+04 -2.024 0.043086 \*   
## hr 2.593e+04 7.865e+03 3.297 0.000991 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1854000 on 2496 degrees of freedom  
## Multiple R-squared: 0.5089, Adjusted R-squared: 0.5042   
## F-statistic: 107.8 on 24 and 2496 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 2521 samples  
## 17 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2268, 2270, 2268, 2269, 2269, 2270, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 1860173 0.5013347 999.7321 1428699 -23.77734 199.8688  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.034 32.550 74.400 197.900 226.500 4854.000

# Random forest for all players

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 3115720 0.5697811 -40965.90 2013343 -165.1337 186.4659

# 9 2918662 0.6021904 -63870.48 1814543 -110.4661 134.3067

# 16 2934899 0.5972576 -63311.40 1815671 -106.9761 131.9021

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(pAllRfmodelError$pred.err.percent.pAllRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0154 11.1900 25.2300 56.4500 69.8200 1070.0000

pAllRfmodelError=buildAndRunRfModel("Avg.Annual","pAllRfmodel",posAllR,modelCtrlRF)

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.3.3

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

## [1] "Year" "hasFreeAgentStatus" "woba"   
## [4] "rbi" "sb" "bb"   
## [7] "r" "single" "sh"   
## [10] "double" "age30to35" "h"   
## [13] "po" "ab" "hr"   
## [16] "g\_idp"   
## Random Forest   
##   
## 2744 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 2192, 2196, 2196, 2196, 2196   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 3095264 0.5744337 -66984.18 2025055 -168.8977 190.1697  
## 9 2899008 0.6068684 -78056.59 1817487 -114.0353 137.5192  
## 16 2906387 0.6045702 -67161.76 1811017 -108.3313 133.0212  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(pAllRfmodelError$pred.err.percent.pAllRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0074 11.2300 25.5400 57.0300 68.9800 1040.0000

# rANDOM FOREST removed outliers for all players

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 1876963 0.5129914 -20300.04 1355609 -136.67975 159.5156

# 9 1795690 0.5365515 -26455.85 1239017 -99.33395 123.2886

# 16 1804260 0.5311841 -21214.53 1237537 -94.85463 119.7167

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(error.rf.model.posAllR.removed.outliers$pred.err.percent.pAll.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0048 10.6800 23.6900 50.4700 64.5500 794.2000

error.rf.model.posAllR.removed.outliers=buildAndRunRfModel("Avg.Annual","pAll.rf.removed.outliers",posAllR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "g"   
## [4] "age30to35" "rbi" "hr"   
## [7] "bb" "single" "woba"   
## [10] "double" "ab" "h"   
## [13] "r" "g\_idp" "sf"   
## [16] "isAllStar"   
## Random Forest   
##   
## 2521 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 2017, 2018, 2016, 2016, 2017   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 1874091 0.5148381 -18434.10 1355646 -136.29041 159.1244  
## 9 1790121 0.5398308 -34066.72 1237857 -98.39626 122.3567  
## 16 1792637 0.5383221 -30262.61 1232009 -94.23042 119.1368  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(error.rf.model.posAllR.removed.outliers$pred.err.percent.pAll.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.02 10.91 23.85 51.12 63.69 961.80

# LOg Model with all players

# RMSE Rsquared ME MAE MPE MAPE

# 0.7515036 0.7126979 0.000634206 0.6021686 -0.2748782 4.28641

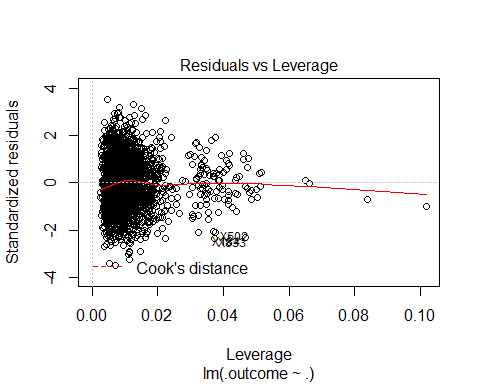
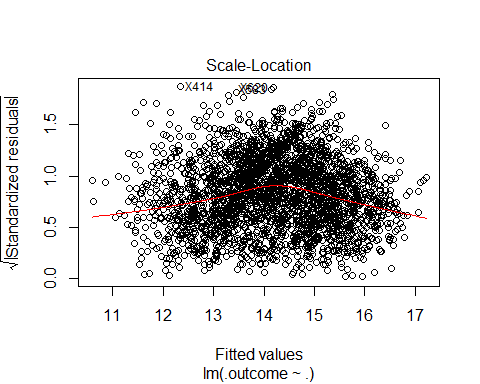
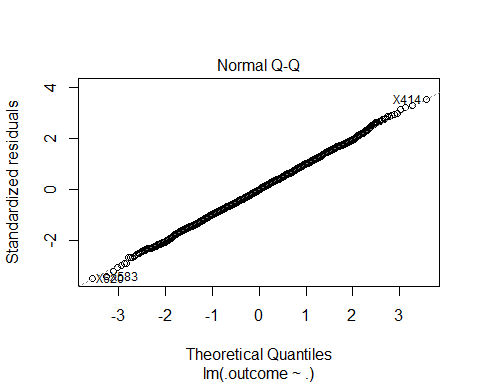
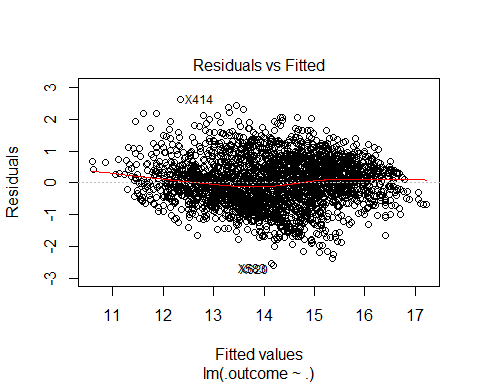
# > summary(pAllLogModelError$pred.err.percent.pAllLogModel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.001651 1.739000 3.570000 4.240000 6.007000 22.360000

full=lm(log(Avg.Annual)~.,data=posAllR)  
null=lm(log(Avg.Annual)~1,data=posAllR)  
  
pAllLogModelError=buildAndRunLogModel(full,null,"Avg.Annual","pAllLogModel",posAllR,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "rbi"   
## [4] "ageUnder25" "bb" "g"   
## [7] "ab" "age25to30" "pos"   
## [10] "isAllStar" "triple" "po"   
## [13] "sh" "ibb" "so"   
## [16] "double" "a" "woba"   
## [19] "g\_idp" "h" "isAwardWinner"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.59380 -0.49882 -0.00729 0.49253 2.62669   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.289e+02 3.843e+00 -33.528 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 1.060e+00 4.358e-02 24.315 < 2e-16 \*\*\*  
## Year 7.055e-02 1.906e-03 37.007 < 2e-16 \*\*\*  
## rbi 3.857e-03 1.279e-03 3.016 0.002585 \*\*   
## ageUnder25TRUE -1.014e+00 7.135e-02 -14.215 < 2e-16 \*\*\*  
## bb 6.613e-03 1.102e-03 6.003 2.20e-09 \*\*\*  
## g -1.874e-02 1.141e-03 -16.423 < 2e-16 \*\*\*  
## ab 7.914e-03 7.678e-04 10.308 < 2e-16 \*\*\*  
## age25to30TRUE -3.055e-01 4.284e-02 -7.133 1.26e-12 \*\*\*  
## pos2B 8.432e-02 8.425e-02 1.001 0.317026   
## pos3B 1.804e-01 8.042e-02 2.243 0.024965 \*   
## posC -1.395e-01 5.559e-02 -2.509 0.012155 \*   
## posCF 3.098e-01 1.609e-01 1.925 0.054301 .   
## posLF 1.774e-01 1.408e-01 1.260 0.207615   
## posOF 3.596e-01 5.904e-02 6.091 1.28e-09 \*\*\*  
## posRF 5.410e-01 1.365e-01 3.963 7.58e-05 \*\*\*  
## posSS 3.204e-01 9.200e-02 3.482 0.000505 \*\*\*  
## isAllStarTRUE 2.597e-01 5.001e-02 5.193 2.22e-07 \*\*\*  
## triple -2.771e-02 7.629e-03 -3.632 0.000286 \*\*\*  
## po 3.465e-04 8.394e-05 4.128 3.76e-05 \*\*\*  
## sh -2.311e-02 5.749e-03 -4.020 5.99e-05 \*\*\*  
## ibb 9.649e-03 4.220e-03 2.287 0.022290 \*   
## so -3.032e-03 7.944e-04 -3.816 0.000138 \*\*\*  
## double -7.651e-03 3.090e-03 -2.476 0.013338 \*   
## a 3.732e-04 2.500e-04 1.493 0.135565   
## woba 2.867e+00 6.833e-01 4.196 2.81e-05 \*\*\*  
## g\_idp 8.635e-03 3.894e-03 2.218 0.026670 \*   
## h -8.998e-03 2.172e-03 -4.144 3.52e-05 \*\*\*  
## isAwardWinnerTRUE 1.063e-01 5.066e-02 2.098 0.035991 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7482 on 2715 degrees of freedom  
## Multiple R-squared: 0.7163, Adjusted R-squared: 0.7134   
## F-statistic: 244.9 on 28 and 2715 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 2744 samples  
## 21 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2469, 2470, 2470, 2469, 2470, 2469, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 0.7508035 0.7120688 0.0004778857 0.6014119 -0.275062 4.280347  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(pAllLogModelError$pred.err.percent.pAllLogModel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.001651 1.739000 3.570000 4.240000 6.007000 22.360000

# log model with all players outliers removed

# RMSE Rsquared ME MAE MPE MAPE

# 0.737453 0.6579692 -0.0008227853 0.5903064 -0.2837138 4.250062

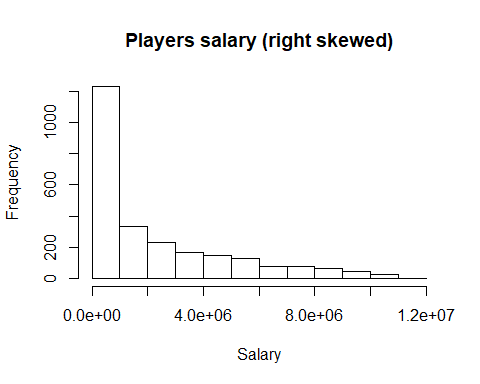
# Tuning parameter 'intercept' was held constant at a value of TRUE

# > summary(error.logmodel.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

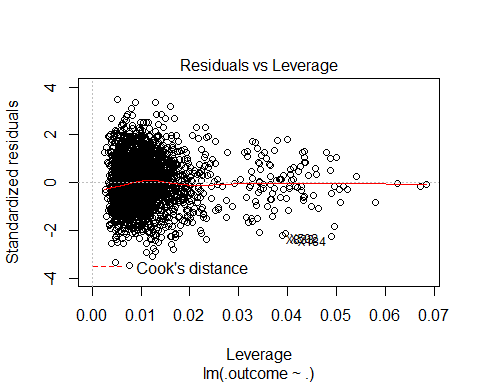
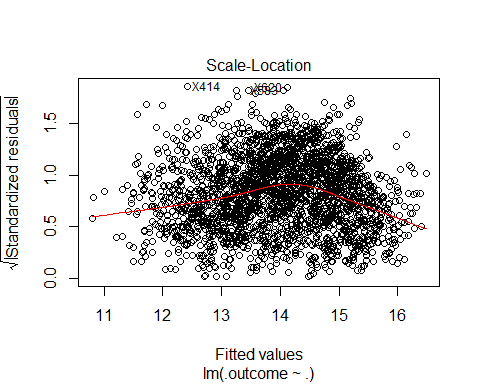
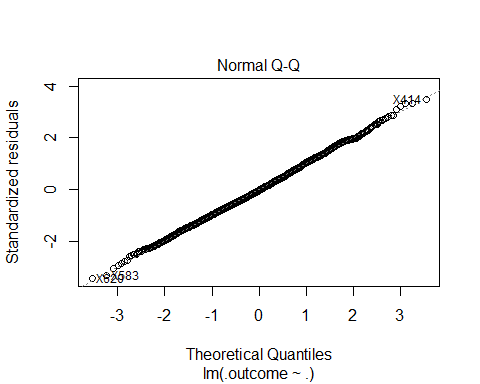
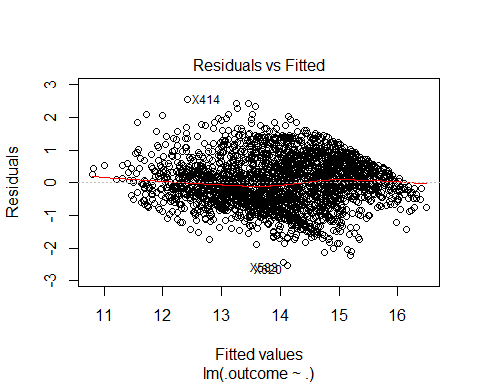
# 0.001393 1.715000 3.539000 4.207000 6.037000 21.700000

# Salary is right skewed, qqplot from the linear regression model shows the residuals are heavy-tailed  
hist(posAllR.removed.outliers$Avg.Annual,xlab="Salary",main="Players salary (right skewed)")



full=lm(log(Avg.Annual)~.,data=posAllR.removed.outliers)  
null=lm(log(Avg.Annual)~1,data=posAllR.removed.outliers)  
error.logmodel.posAllR.removed.outliers=buildAndRunLogModel(full,null,"Avg.Annual","pAll.removed.outliers",posAllR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "rbi"   
## [4] "ageUnder25" "g" "ab"   
## [7] "bb" "age25to30" "isAllStar"   
## [10] "pos" "po" "sh"   
## [13] "so" "h" "woba"   
## [16] "isAwardWinner" "triple" "ibb"   
## [19] "double" "a"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.51678 -0.49301 -0.02435 0.49423 2.54339   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.158e+02 3.978e+00 -29.121 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 9.999e-01 4.428e-02 22.584 < 2e-16 \*\*\*  
## Year 6.415e-02 1.971e-03 32.544 < 2e-16 \*\*\*  
## rbi 4.452e-03 1.312e-03 3.393 0.000702 \*\*\*  
## ageUnder25TRUE -9.736e-01 7.112e-02 -13.689 < 2e-16 \*\*\*  
## g -1.673e-02 1.146e-03 -14.591 < 2e-16 \*\*\*  
## ab 7.493e-03 7.909e-04 9.474 < 2e-16 \*\*\*  
## bb 6.217e-03 1.137e-03 5.466 5.05e-08 \*\*\*  
## age25to30TRUE -2.836e-01 4.400e-02 -6.445 1.38e-10 \*\*\*  
## isAllStarTRUE 2.561e-01 5.382e-02 4.759 2.06e-06 \*\*\*  
## pos2B 7.721e-02 8.470e-02 0.912 0.362046   
## pos3B 1.424e-01 8.220e-02 1.732 0.083381 .   
## posC -1.139e-01 5.686e-02 -2.004 0.045228 \*   
## posCF 3.019e-01 1.619e-01 1.865 0.062358 .   
## posLF 2.089e-01 1.413e-01 1.479 0.139354   
## posOF 3.225e-01 6.099e-02 5.288 1.34e-07 \*\*\*  
## posRF 5.718e-01 1.388e-01 4.121 3.89e-05 \*\*\*  
## posSS 2.575e-01 9.331e-02 2.760 0.005824 \*\*   
## po 3.235e-04 9.012e-05 3.590 0.000338 \*\*\*  
## sh -2.206e-02 5.778e-03 -3.818 0.000138 \*\*\*  
## so -3.436e-03 8.117e-04 -4.233 2.39e-05 \*\*\*  
## h -8.850e-03 2.266e-03 -3.906 9.62e-05 \*\*\*  
## woba 2.017e+00 6.987e-01 2.887 0.003919 \*\*   
## isAwardWinnerTRUE 1.302e-01 5.322e-02 2.446 0.014502 \*   
## triple -2.079e-02 7.581e-03 -2.742 0.006141 \*\*   
## ibb 1.011e-02 5.011e-03 2.017 0.043812 \*   
## double -6.197e-03 3.231e-03 -1.918 0.055231 .   
## a 4.869e-04 2.536e-04 1.920 0.054965 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7344 on 2493 degrees of freedom  
## Multiple R-squared: 0.664, Adjusted R-squared: 0.6604   
## F-statistic: 182.5 on 27 and 2493 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 2521 samples  
## 20 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2269, 2269, 2269, 2269, 2269, 2270, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 0.7370128 0.6585368 0.00065035 0.5900884 -0.2732366 4.247975  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.logmodel.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers)

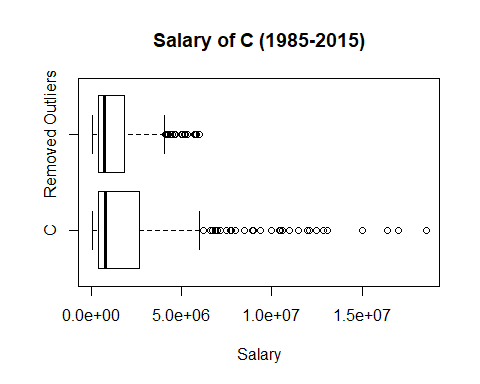
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.001393 1.715000 3.539000 4.207000 6.037000 21.700000

# Populate data for catchers

regressionFeatures=c("Avg.Annual","Year","woba","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
posCR=winningBatters[which(winningBatters$pos=="C"),]  
posCR=posCR[,regressionFeatures]

# Show boxplot of salaries and determine if there are outliers

posCR.removed.outliers=removeOutliers(posCR,"Avg.Annual")  
boxplot(list(posCR$Avg.Annual,posCR.removed.outliers$Avg.Annual),xlab="Salary",names=c("C","Removed Outliers"),main="Salary of C (1985-2015)",horizontal=TRUE)



# Normal model for catchers

# RMSE Rsquared ME MAE MPE MAPE

# 2314594 0.5167817 3726.043 1628863 26.66264 266.1617

# Tuning parameter 'intercept' was held constant at a value of TRUE

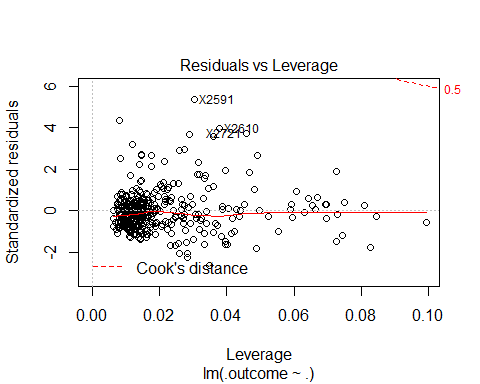
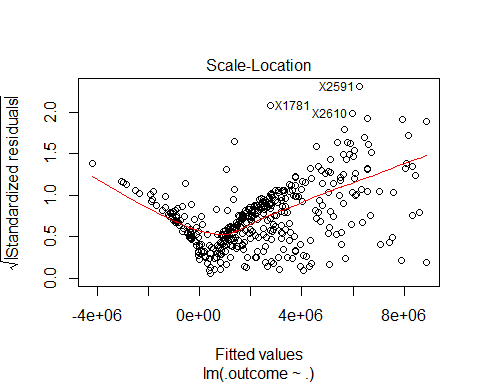
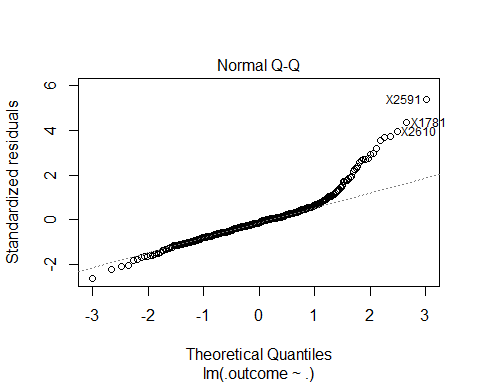
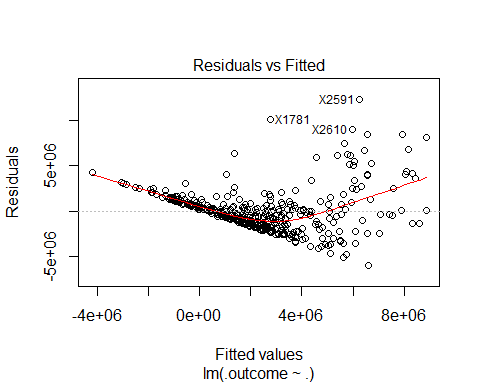
# > summary(pCModel1Error$pred.err.percent.pCModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.455 34.490 109.900 259.400 266.800 5496.000

full=lm(Avg.Annual~.,data=posCR)  
null=lm(Avg.Annual~1,data=posCR)  
pCModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","pCModel1",posCR,modelCtrlLM)

## [1] "rbi" "hasFreeAgentStatus" "Year"   
## [4] "sh" "r" "ageUnder25"   
## [7] "isAwardWinner"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5978444 -1336207 -321830 744474 12293765   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -286580000 29326258 -9.772 < 2e-16 \*\*\*  
## rbi 28421 12820 2.217 0.02723 \*   
## hasFreeAgentStatusTRUE 2455180 247301 9.928 < 2e-16 \*\*\*  
## Year 142483 14628 9.740 < 2e-16 \*\*\*  
## sh -125083 47821 -2.616 0.00926 \*\*   
## r 36456 14403 2.531 0.01178 \*   
## ageUnder25TRUE -1173686 586767 -2.000 0.04619 \*   
## isAwardWinnerTRUE 726859 432460 1.681 0.09364 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2330000 on 377 degrees of freedom  
## Multiple R-squared: 0.5191, Adjusted R-squared: 0.5102   
## F-statistic: 58.15 on 7 and 377 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 385 samples  
## 7 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 348, 345, 346, 345, 347, 346, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 2301389 0.5341136 3001.393 1618692 26.09895 266.0335  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(pCModel1Error$pred.err.percent.pCModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.455 34.490 109.900 259.400 266.800 5496.000

# With salary outliers removed for catchers

# RMSE Rsquared ME MAE MPE MAPE

# 1049067 0.4059886 8354.375 798137 -31.50746 122.6718

# Tuning parameter 'intercept' was held constant at a value of TRUE

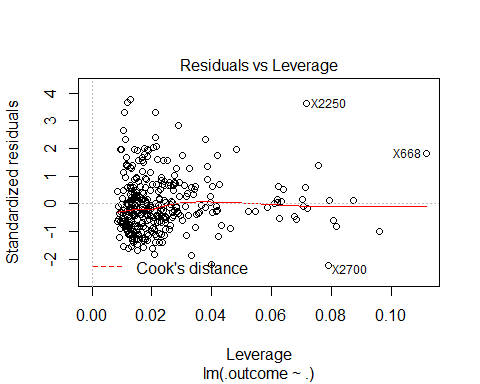
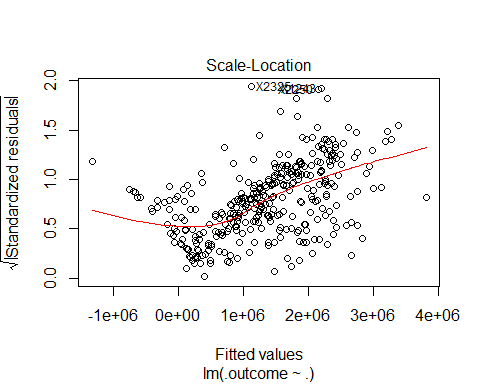
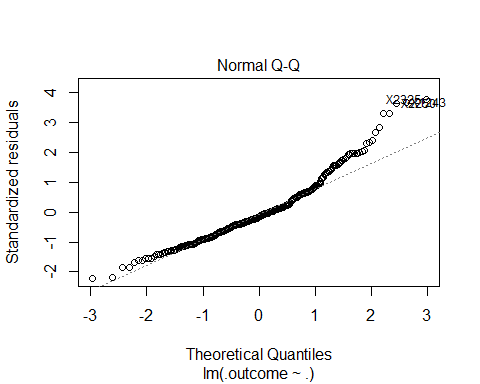
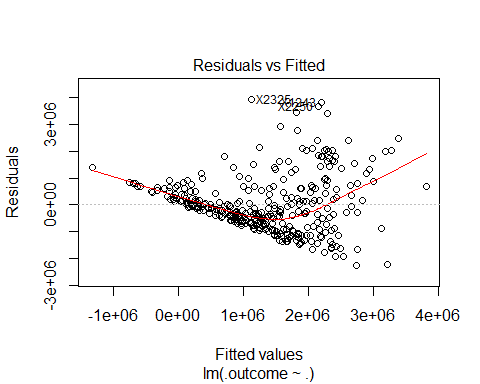
# > summary(error.model.posCR.removed.outliers$pred.err.percent.pC.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0614 31.2000 62.4000 118.6000 154.5000 1808.0000

full=lm(Avg.Annual~.,data=posCR.removed.outliers)  
null=lm(Avg.Annual~1,data=posCR.removed.outliers)  
error.model.posCR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","pC.removed.outliers",posCR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "po" "Year"   
## [4] "r" "ageUnder25" "age35to50"   
## [7] "triple"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2234387 -674976 -138332 518728 3910977   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.159e+08 1.426e+07 -8.130 8.44e-15 \*\*\*  
## hasFreeAgentStatusTRUE 1.283e+06 1.262e+05 10.173 < 2e-16 \*\*\*  
## po 7.351e+02 4.116e+02 1.786 0.07501 .   
## Year 5.779e+04 7.116e+03 8.120 9.00e-15 \*\*\*  
## r 2.109e+04 5.418e+03 3.892 0.00012 \*\*\*  
## ageUnder25TRUE -6.672e+05 2.630e+05 -2.537 0.01162 \*   
## age35to50TRUE -3.603e+05 1.779e+05 -2.026 0.04360 \*   
## triple -1.041e+05 5.733e+04 -1.817 0.07018 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1045000 on 335 degrees of freedom  
## Multiple R-squared: 0.4179, Adjusted R-squared: 0.4057   
## F-statistic: 34.36 on 7 and 335 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 343 samples  
## 7 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 310, 309, 309, 310, 307, 310, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 1045433 0.4137854 4353.718 798108.6 -31.22627 123.6549  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.posCR.removed.outliers$pred.err.percent.pC.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0614 31.2000 62.4000 118.6000 154.5000 1808.0000

# Random forest for catchers

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 2413951 0.4953062 -28468.04 1504535 -167.7582 190.1735

# 9 2240692 0.5602224 -52929.95 1371180 -136.0973 158.5622

# 16 2187894 0.5738195 -50625.89 1333821 -128.8662 151.8840

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(pCRfmodelError$pred.err.percent.pCRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0995 10.7200 22.8500 57.0700 58.1000 2244.0000

pCRfmodelError=buildAndRunRfModel("Avg.Annual","pCRfmodel",posCR,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "rbi"   
## [4] "bb" "r" "ab"   
## [7] "so" "h" "po"   
## [10] "age25to30" "hbp" "single"   
## [13] "hr" "double" "g"   
## [16] "a"   
## Random Forest   
##   
## 385 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 309, 308, 309, 306, 308   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 2407570 0.5045307 37717.35 1465763 -154.0303 175.9677  
## 9 2289515 0.5373857 17821.12 1360271 -124.2413 146.4787  
## 16 2305669 0.5226897 25374.41 1369084 -116.2442 140.2063  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(pCRfmodelError$pred.err.percent.pCRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.097 12.900 24.380 62.260 70.120 2470.000

# rANDOM FOREST removed outliers for catchers

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 1151180 0.2930748 -2801.201 845520.7 -109.94985 134.5661

# 9 1121897 0.3150358 -6071.415 805970.2 -88.89415 114.2926

# 16 1122092 0.3147121 -3561.572 795732.3 -82.24680 108.6876

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(error.rf.model.posCR.removed.outliers$pred.err.percent.pC.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.1562 12.0100 23.4100 45.7900 55.6100 1026.0000

error.rf.model.posCR.removed.outliers=buildAndRunRfModel("Avg.Annual","pC.rf.removed.outliers",posCR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "po" "age30to35"   
## [4] "Year" "so" "age25to30"   
## [7] "ab" "g\_idp" "h"   
## [10] "single" "r" "sf"   
## [13] "rbi" "a" "bb"   
## [16] "g"   
## Random Forest   
##   
## 343 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 273, 275, 275, 274, 275   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 1135504 0.3209767 -19151.70 833012.4 -109.65325 133.4349  
## 9 1093005 0.3596168 -25971.44 777817.6 -86.84654 110.8769  
## 16 1093971 0.3636345 -26354.91 772734.3 -80.63081 105.8508  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(error.rf.model.posCR.removed.outliers$pred.err.percent.pC.rf.removed.outliers)

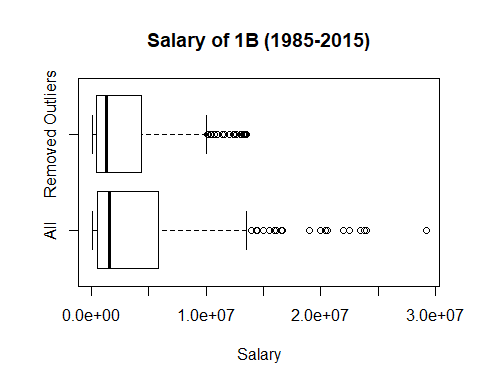
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0031 10.9400 22.5600 44.1400 51.6200 862.8000

# setup data for 1B players

regressionFeatures=c("Avg.Annual","Year","woba","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
pos1BR=winningBatters[which(winningBatters$pos=="1B"),]  
pos1BR=pos1BR[,regressionFeatures]

# Show boxplot of salaries and determine if there are outliers

pos1BR.removed.outliers=removeOutliers(pos1BR,"Avg.Annual")  
boxplot(list(pos1BR$Avg.Annual,pos1BR.removed.outliers$Avg.Annual),xlab="Salary",names=c("All","Removed Outliers"),main="Salary of 1B (1985-2015)",horizontal=TRUE)



# Normal model for 1B

# RMSE Rsquared ME MAE MPE MAPE

# 3893143 0.5164647 -158.9451 2905085 47.43024 391.6925

# Tuning parameter 'intercept' was held constant at a value of TRUE

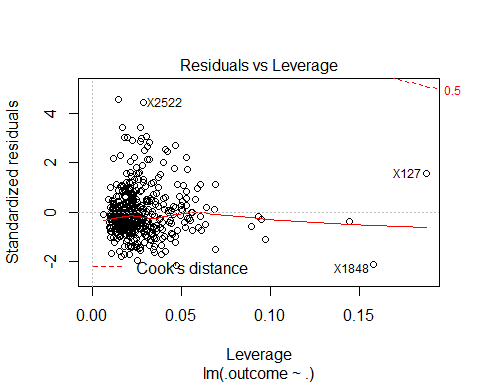
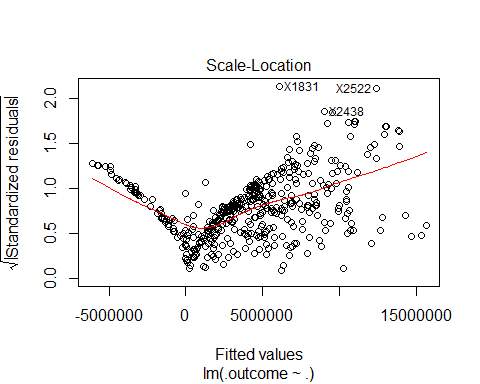
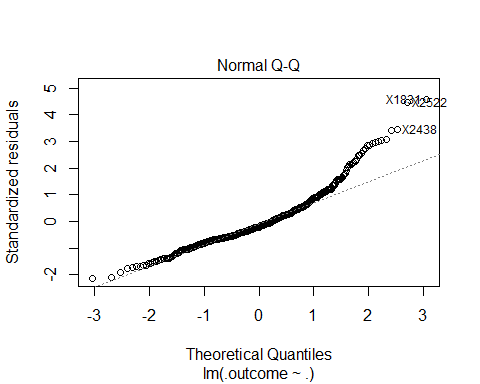
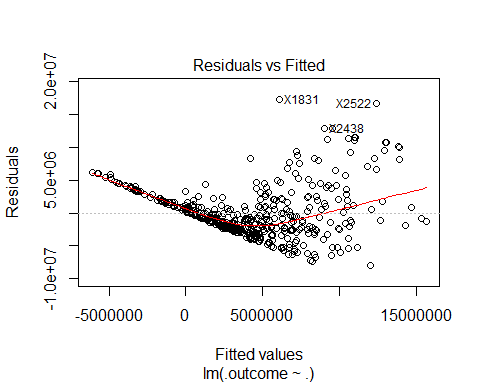
# > summary(p1BModel1Error$pred.err.percent.p1BModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.47 36.84 97.50 374.40 349.80 9865.00

full=lm(Avg.Annual~.,data=pos1BR)  
null=lm(Avg.Annual~1,data=pos1BR)  
p1BModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","p1BModel1",pos1BR,modelCtrlLM)

## [1] "ab" "Year" "hasFreeAgentStatus"  
## [4] "g" "ibb" "isAllStar"   
## [7] "h" "woba" "g\_idp"   
## [10] "so" "cs"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7956202 -2374165 -768995 1647985 17305845   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -533231707 50317333 -10.597 < 2e-16 \*\*\*  
## ab 49683 8563 5.802 1.28e-08 \*\*\*  
## Year 262835 24868 10.569 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 3685799 383578 9.609 < 2e-16 \*\*\*  
## g -62906 12590 -4.996 8.55e-07 \*\*\*  
## ibb 105863 45732 2.315 0.02109 \*   
## isAllStarTRUE 2200941 644566 3.415 0.00070 \*\*\*  
## h -101790 25388 -4.009 7.19e-05 \*\*\*  
## woba 21534447 6703288 3.213 0.00142 \*\*   
## g\_idp 96814 51563 1.878 0.06112 .   
## so -15986 9110 -1.755 0.08001 .   
## cs -189909 112200 -1.693 0.09126 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3821000 on 426 degrees of freedom  
## Multiple R-squared: 0.5419, Adjusted R-squared: 0.5301   
## F-statistic: 45.82 on 11 and 426 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 438 samples  
## 11 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 394, 395, 394, 394, 394, 395, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 3904933 0.517908 -17941.84 2903638 51.96675 393.3183  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(p1BModel1Error$pred.err.percent.p1BModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.47 36.84 97.50 374.40 349.80 9865.00

# With salary outliers removed for 1B

# RMSE Rsquared ME MAE MPE MAPE

# 2471295 0.4879897 12358.65 1900732 -11.51971 252.6585

# Tuning parameter 'intercept' was held constant at a value of TRUE

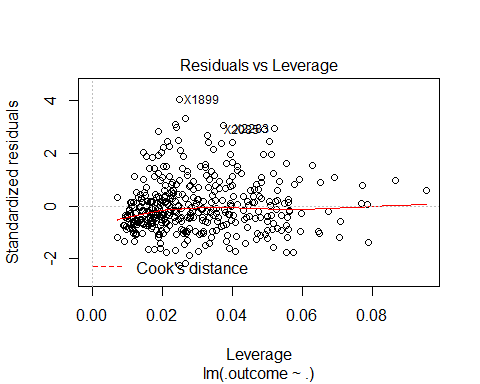
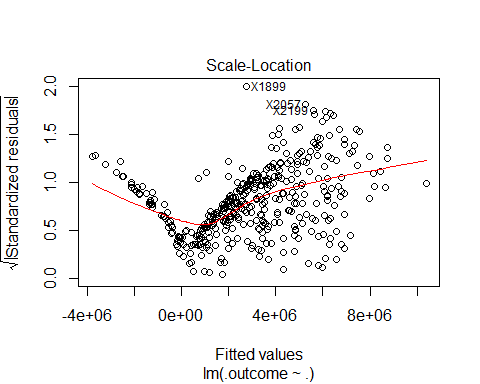
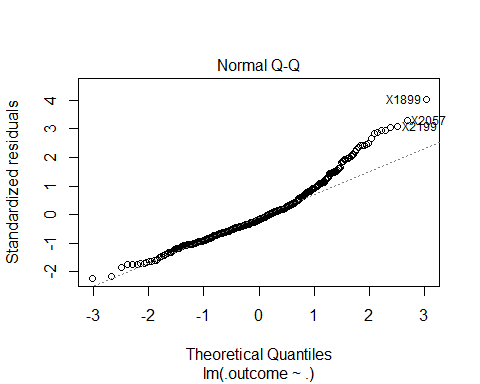
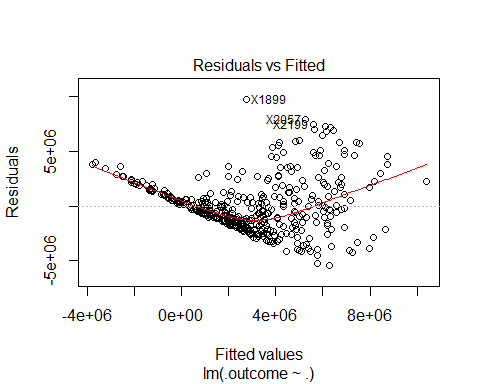
# > summary(error.model.pos1BR.removed.outliers$pred.err.percent.1B.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.239 32.790 81.510 241.600 276.400 6159.000

full=lm(Avg.Annual~.,data=pos1BR.removed.outliers)  
null=lm(Avg.Annual~1,data=pos1BR.removed.outliers)  
error.model.pos1BR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","1B.removed.outliers",pos1BR.removed.outliers,modelCtrlLM)

## [1] "bb" "hasFreeAgentStatus" "Year"   
## [4] "hr" "g" "ab"   
## [7] "isAllStar" "g\_idp" "ageUnder25"   
## [10] "age25to30" "rbi"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5416562 -1572628 -428337 1071188 9746240   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -294119180 30842565 -9.536 < 2e-16 \*\*\*  
## bb 31447 7612 4.131 4.41e-05 \*\*\*  
## hasFreeAgentStatusTRUE 1694937 407207 4.162 3.87e-05 \*\*\*  
## Year 147985 15396 9.612 < 2e-16 \*\*\*  
## hr 73987 27857 2.656 0.00823 \*\*   
## g -52603 8216 -6.403 4.35e-10 \*\*\*  
## ab 12569 2898 4.337 1.84e-05 \*\*\*  
## isAllStarTRUE 1101671 431539 2.553 0.01106 \*   
## g\_idp 62691 34457 1.819 0.06961 .   
## ageUnder25TRUE -1796079 636093 -2.824 0.00499 \*\*   
## age25to30TRUE -947561 407614 -2.325 0.02060 \*   
## rbi -26431 14479 -1.825 0.06869 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2454000 on 393 degrees of freedom  
## Multiple R-squared: 0.5111, Adjusted R-squared: 0.4974   
## F-statistic: 37.35 on 11 and 393 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 405 samples  
## 11 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 365, 365, 363, 365, 365, 365, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 2480337 0.4918589 17586.6 1898084 -12.02343 249.6309  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.pos1BR.removed.outliers$pred.err.percent.1B.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.239 32.790 81.510 241.600 276.400 6159.000

# Random forest for 1B

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 3857420 0.5474452 -66393.70 2680325 -241.9065 264.9550

# 9 3395541 0.6408556 -82835.05 2304728 -180.4462 202.7582

# 16 3428451 0.6245384 -67236.27 2286221 -164.7468 188.1807

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(p1BRfmodelError$pred.err.percent.p1BRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0051 11.6300 28.6100 79.6600 88.4400 1012.0000

p1BRfmodelError=buildAndRunRfModel("Avg.Annual","p1BRfmodel",pos1BR,modelCtrlRF)

## [1] "Year" "hasFreeAgentStatus" "po"   
## [4] "bb" "a" "hr"   
## [7] "ab" "r" "h"   
## [10] "rbi" "sh" "g\_idp"   
## [13] "so" "g" "hbp"   
## [16] "sf"   
## Random Forest   
##   
## 438 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 350, 350, 350, 351, 351   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 3884332 0.5446401 -98505.40 2698887 -253.5430 276.0613  
## 9 3473274 0.6207988 -93818.56 2360052 -190.5030 212.6931  
## 16 3471871 0.6154472 -62425.90 2331409 -171.9053 194.7968  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(p1BRfmodelError$pred.err.percent.p1BRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1489 10.7500 27.3800 72.4000 84.2700 853.3000

# rANDOM FOREST removed outliers for 1B

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 2605671 0.4577912 -30563.71 1875753 -193.7624 216.6985

# 9 2466527 0.5007996 -35342.91 1729828 -152.0031 174.6840

# 16 2465291 0.4963060 -22642.68 1702055 -137.4516 160.9274

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(error.rf.model.pos1BR.removed.outliers$pred.err.percent.p1B.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0183 11.3800 28.2800 64.7200 74.2400 911.6000

error.rf.model.pos1BR.removed.outliers=buildAndRunRfModel("Avg.Annual","p1B.rf.removed.outliers",pos1BR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "bb"   
## [4] "isAllStar" "single" "rbi"   
## [7] "ab" "po" "so"   
## [10] "r" "h" "g"   
## [13] "hr" "age30to35" "g\_idp"   
## [16] "a"   
## Random Forest   
##   
## 405 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 323, 326, 324, 323, 324   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 2682350 0.4225717 -10177.07 1923983 -199.2209 223.1343  
## 9 2533314 0.4728603 -49298.57 1779128 -159.9433 182.9417  
## 16 2517372 0.4730117 -59373.39 1730286 -147.4306 170.2640  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(error.rf.model.pos1BR.removed.outliers$pred.err.percent.p1B.rf.removed.outliers)

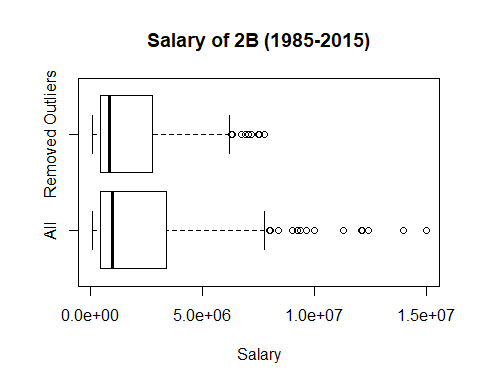
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.05 11.20 27.81 64.18 75.89 1124.00

# setup data for 2B players

regressionFeatures=c("Avg.Annual","Year","woba","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
pos2BR=winningBatters[which(winningBatters$pos=="2B"),]  
pos2BR=pos2BR[,regressionFeatures]

# Show boxplot of salaries and determine if there are outliers

pos2BR.removed.outliers=removeOutliers(pos2BR,"Avg.Annual")  
boxplot(list(pos2BR$Avg.Annual,pos2BR.removed.outliers$Avg.Annual),xlab="Salary",names=c("All","Removed Outliers"),main="Salary of 2B (1985-2015)",horizontal=TRUE)



# Normal model for 2b

# RMSE Rsquared ME MAE MPE MAPE

# 2072290 0.5457813 6027.34 1563885 -15.60563 227.9686

# Tuning parameter 'intercept' was held constant at a value of TRUE

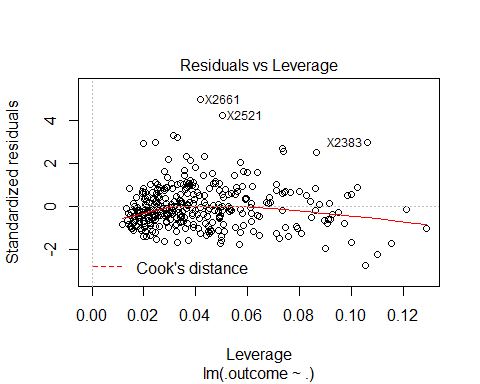
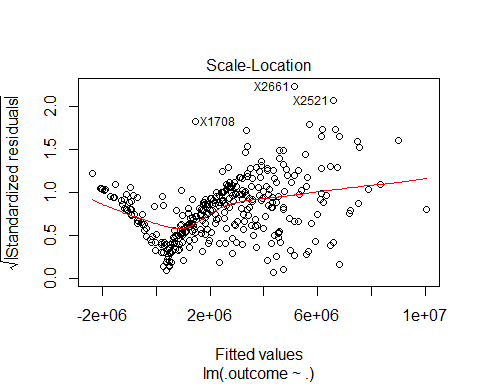
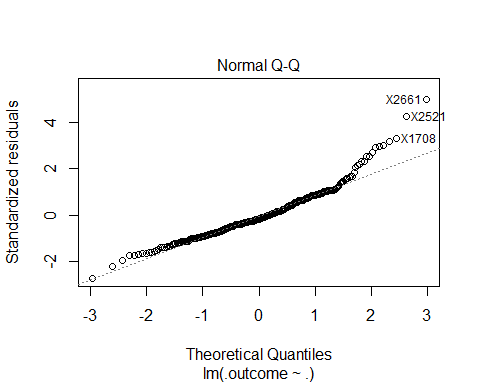
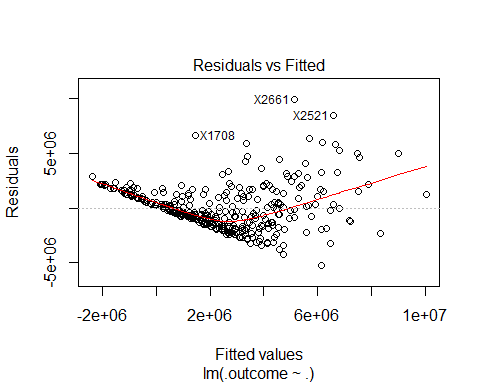
# > summary(p2BModel1Error$pred.err.percent.p2BModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.184 34.270 81.300 216.900 246.000 3192.000

full=lm(Avg.Annual~.,data=pos2BR)  
null=lm(Avg.Annual~1,data=pos2BR)  
p2BModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","p2BModel1",pos2BR,modelCtrlLM)

## [1] "hr" "hasFreeAgentStatus" "Year"   
## [4] "po" "g" "bb"   
## [7] "so" "sh" "e"   
## [10] "isAllStar" "sf" "rbi"   
## [13] "double"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5199960 -1341287 -315795 1101246 9863065   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -260007563 30937535 -8.404 1.36e-15 \*\*\*  
## hr 75196 30640 2.454 0.01465 \*   
## hasFreeAgentStatusTRUE 2156792 228365 9.445 < 2e-16 \*\*\*  
## Year 130474 15459 8.440 1.06e-15 \*\*\*  
## po 12179 2514 4.845 1.96e-06 \*\*\*  
## g -22847 6916 -3.304 0.00106 \*\*   
## bb 32606 7601 4.290 2.36e-05 \*\*\*  
## so -13543 5638 -2.402 0.01686 \*   
## sh -87848 35805 -2.454 0.01467 \*   
## e -59361 31296 -1.897 0.05875 .   
## isAllStarTRUE 627778 394685 1.591 0.11268   
## sf -160196 66105 -2.423 0.01592 \*   
## rbi 31446 14078 2.234 0.02618 \*   
## double -31465 19114 -1.646 0.10069   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2025000 on 325 degrees of freedom  
## Multiple R-squared: 0.5608, Adjusted R-squared: 0.5432   
## F-statistic: 31.92 on 13 and 325 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 339 samples  
## 13 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 305, 305, 306, 305, 305, 304, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 2075506 0.5282377 3119.49 1558616 -15.92077 226.5241  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(p2BModel1Error$pred.err.percent.p2BModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.184 34.270 81.300 216.900 246.000 3192.000

# With salary outliers removed for 2b

# RMSE Rsquared ME MAE MPE MAPE

# 1405574 0.518049 12390.53 1112121 -27.75935 168.8912

# Tuning parameter 'intercept' was held constant at a value of TRUE

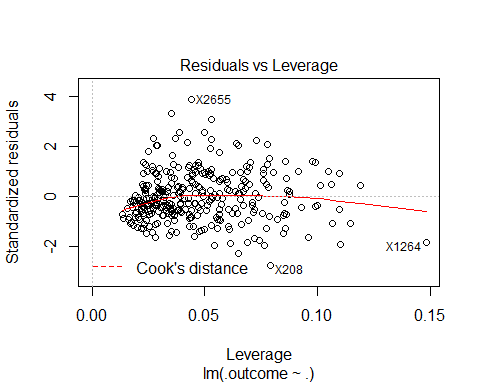
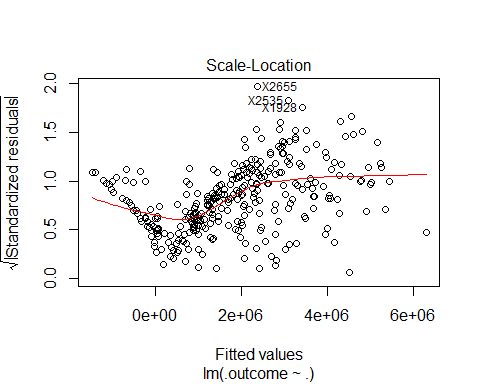
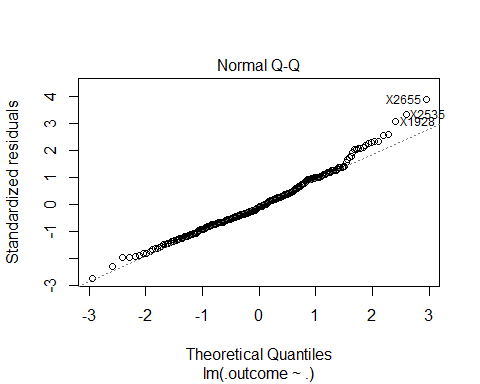
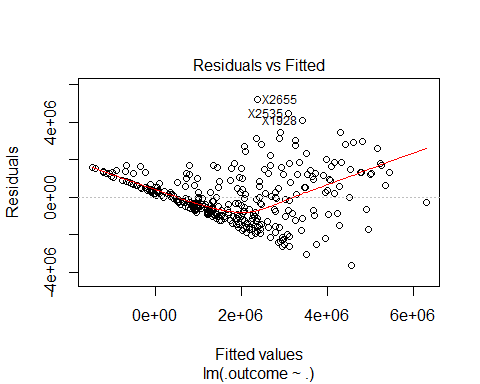
# > summary(error.model.pos2BR.removed.outliers$pred.err.percent.p2B.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0962 31.1600 73.3000 161.1000 186.2000 1587.0000

full=lm(Avg.Annual~.,data=pos2BR.removed.outliers)  
null=lm(Avg.Annual~1,data=pos2BR.removed.outliers)  
error.model.pos2BR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","p2B.removed.outliers",pos2BR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "g"   
## [4] "po" "bb" "isAwardWinner"   
## [7] "hr" "so" "sh"   
## [10] "cs" "e" "age30to35"   
## [13] "ab" "ageUnder25"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3608746 -893633 -144779 814077 5201059   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -171568803 21714919 -7.901 5.25e-14 \*\*\*  
## hasFreeAgentStatusTRUE 1247811 195510 6.382 6.58e-10 \*\*\*  
## Year 86342 10842 7.964 3.45e-14 \*\*\*  
## g -24922 6412 -3.887 0.000125 \*\*\*  
## po 5837 2044 2.855 0.004597 \*\*   
## bb 26491 5316 4.983 1.06e-06 \*\*\*  
## isAwardWinnerTRUE 603106 287745 2.096 0.036919 \*   
## hr 74667 16288 4.584 6.69e-06 \*\*\*  
## so -13353 4108 -3.250 0.001284 \*\*   
## sh -36106 24553 -1.471 0.142460   
## cs 56738 27084 2.095 0.037013 \*   
## e -43469 22968 -1.893 0.059372 .   
## age30to35TRUE 294837 192122 1.535 0.125923   
## ab 2953 1866 1.582 0.114687   
## ageUnder25TRUE -484254 321871 -1.505 0.133501   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1366000 on 301 degrees of freedom  
## Multiple R-squared: 0.5595, Adjusted R-squared: 0.539   
## F-statistic: 27.31 on 14 and 301 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 316 samples  
## 14 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 284, 286, 284, 285, 284, 284, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 1395520 0.5162681 -12422.06 1106385 -29.21147 169.0562  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.pos2BR.removed.outliers$pred.err.percent.p2B.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0962 31.1600 73.3000 161.1000 186.2000 1587.0000

# Random forest for 2b

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 2127317 0.5328158 -40188.66 1524099 -165.2893 189.0960

# 9 1994474 0.5729624 -36399.43 1400455 -132.5753 157.3607

# 16 1962461 0.5812930 -41389.05 1371507 -123.6432 148.9246

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(p2BRfmodelError$pred.err.percent.p2BRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.13 12.02 26.93 59.55 74.49 474.90

p2BRfmodelError=buildAndRunRfModel("Avg.Annual","p2BRfmodel",pos2BR,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "hr"   
## [4] "po" "age30to35" "ab"   
## [7] "a" "rbi" "g\_idp"   
## [10] "g" "hbp" "r"   
## [13] "woba" "double" "age25to30"   
## [16] "h"   
## Random Forest   
##   
## 339 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 271, 270, 271, 272, 272   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 2177227 0.4972417 -66362.01 1515942 -162.4522 185.7949  
## 9 2050094 0.5408958 -71211.09 1400352 -131.1916 155.0275  
## 16 2028780 0.5470657 -64298.70 1366088 -119.8142 144.6814  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(p2BRfmodelError$pred.err.percent.p2BRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2337 10.9000 24.3800 56.5000 68.5800 465.9000

# rANDOM FOREST removed outliers for 2b

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 1519538 0.4514239 -30302.14 1153496 -134.5513 160.3371

# 9 1440565 0.5053734 -23827.30 1051736 -107.6368 132.7564

# 16 1434474 0.5091674 -20279.64 1035138 -98.7905 124.6600

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(error.rf.model.pos2BR.removed.outliers$pred.err.percent.p2B.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.1979 10.5600 23.9800 50.1800 61.6600 462.7000

error.rf.model.pos2BR.removed.outliers=buildAndRunRfModel("Avg.Annual","p2B.rf.removed.outliers",pos2BR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "r"   
## [4] "woba" "hr" "bb"   
## [7] "po" "ab" "double"   
## [10] "g" "rbi" "a"   
## [13] "so" "single" "isAwardWinner"   
## [16] "age25to30"   
## Random Forest   
##   
## 316 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 254, 252, 254, 252, 252   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 1506405 0.4712264 3052.278 1138646 -130.21626 156.1355  
## 9 1404990 0.5232138 -15467.486 1036362 -104.68089 129.9416  
## 16 1403882 0.5186512 -8845.199 1025116 -97.87208 123.9536  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(error.rf.model.pos2BR.removed.outliers$pred.err.percent.p2B.rf.removed.outliers)

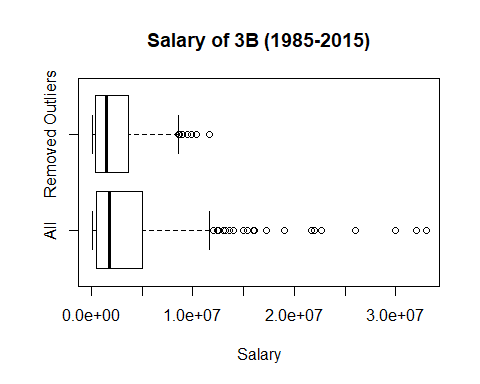
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.028 12.500 25.170 51.240 56.550 499.900

# setup data for 3b players

regressionFeatures=c("Avg.Annual","Year","woba","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
pos3BR=winningBatters[which(winningBatters$pos=="3B"),]  
pos3BR=pos3BR[,regressionFeatures]

# Show boxplot of salaries and determine if there are outliers

pos3BR.removed.outliers=removeOutliers(pos3BR,"Avg.Annual")  
  
boxplot(list(pos3BR$Avg.Annual,pos3BR.removed.outliers$Avg.Annual),xlab="Salary",names=c("All","Removed Outliers"),main="Salary of 3B (1985-2015)",horizontal=TRUE)



# Normal model for 3b

# RMSE Rsquared ME MAE MPE MAPE

# 3991364 0.5146184 -11341.8 2905305 59.92536 393.2048

# Tuning parameter 'intercept' was held constant at a value of TRUE

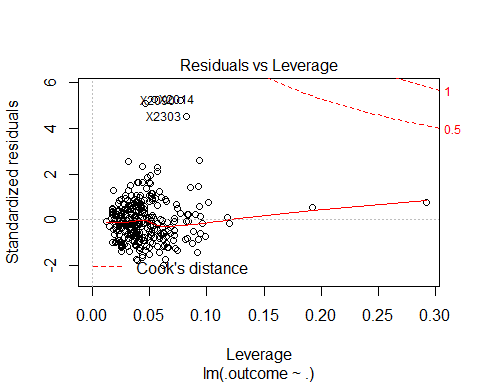
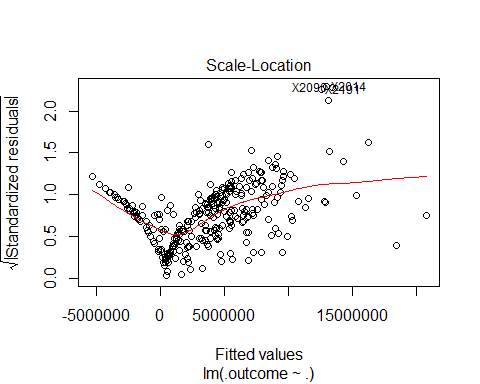
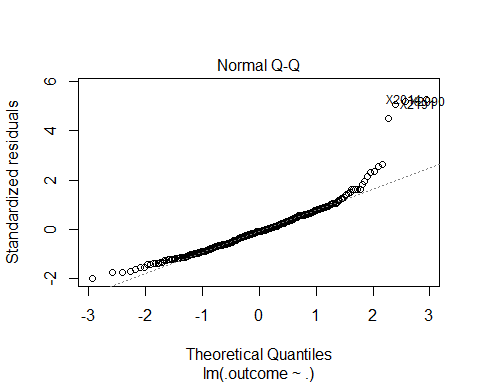
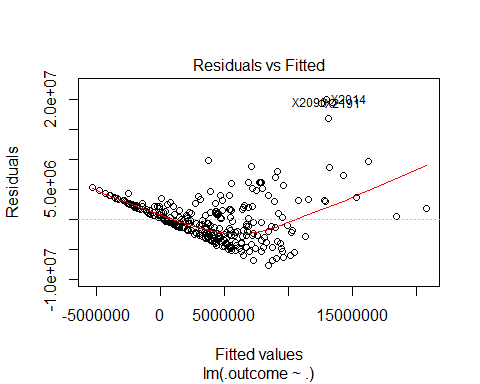
# > summary(p3BModel1Error$pred.err.percent.p3BModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.536 36.230 97.610 367.700 326.800 5989.000

full=lm(Avg.Annual~.,data=pos3BR)  
null=lm(Avg.Annual~1,data=pos3BR)  
p3BModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","p3BModel1",pos3BR,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "isAllStar"   
## [4] "triple" "age35to50" "woba"   
## [7] "a" "rbi" "double"   
## [10] "r" "g" "hbp"   
## [13] "ab"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7534800 -2542654 -332743 1874051 20039430   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -451008526 56309278 -8.009 2.90e-14 \*\*\*  
## hasFreeAgentStatusTRUE 3639541 507014 7.178 6.06e-12 \*\*\*  
## Year 223564 27948 7.999 3.10e-14 \*\*\*  
## isAllStarTRUE 1335389 708893 1.884 0.060607 .   
## triple -432498 149708 -2.889 0.004160 \*\*   
## age35to50TRUE 1222425 699651 1.747 0.081674 .   
## woba 15525014 8506051 1.825 0.069014 .   
## a -16835 5548 -3.034 0.002631 \*\*   
## rbi 61671 18013 3.424 0.000708 \*\*\*  
## double -155580 49928 -3.116 0.002018 \*\*   
## r 45455 25508 1.782 0.075806 .   
## g -52581 18056 -2.912 0.003871 \*\*   
## hbp 158088 86339 1.831 0.068137 .   
## ab 15319 7395 2.072 0.039204 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3915000 on 287 degrees of freedom  
## Multiple R-squared: 0.5357, Adjusted R-squared: 0.5146   
## F-statistic: 25.47 on 13 and 287 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 301 samples  
## 13 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 271, 270, 270, 273, 271, 270, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 3903821 0.490948 32029.52 2843821 62.39741 389.3434  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(p3BModel1Error$pred.err.percent.p3BModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.536 36.230 97.610 367.700 326.800 5989.000

# With salary outliers removed for 3b

# RMSE Rsquared ME MAE MPE MAPE

# 1886593 0.4670419 -9163.871 1467069 -23.95106 203.5158

# Tuning parameter 'intercept' was held constant at a value of TRUE

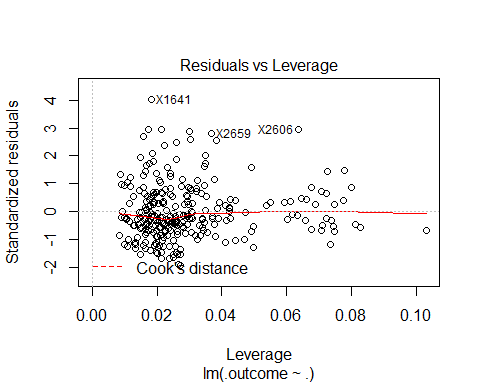
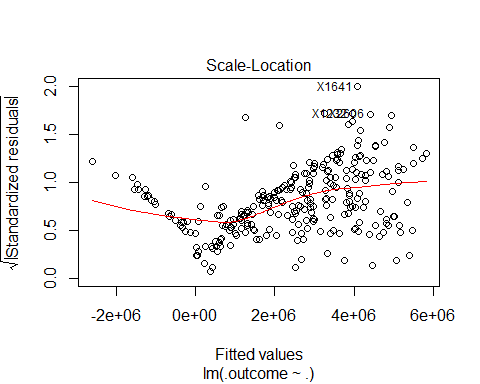
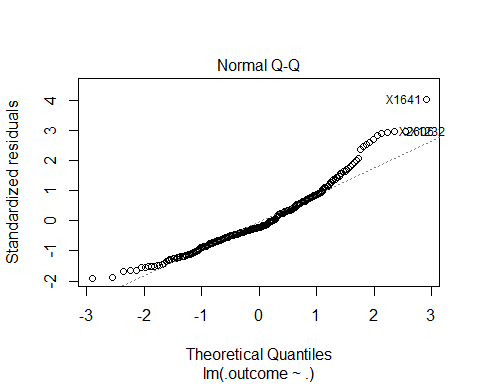
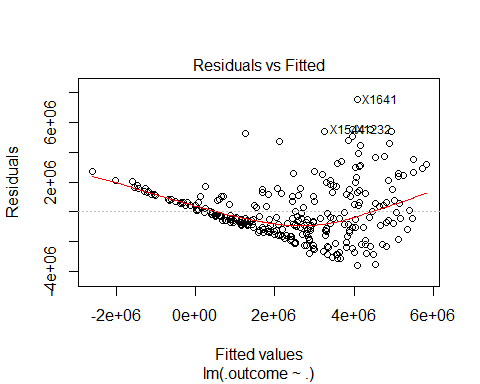
# > summary(error.model.pos3BR.removed.outliers$pred.err.percent.p3B.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.817 33.020 65.310 198.400 220.300 3261.000

full=lm(Avg.Annual~.,data=pos3BR.removed.outliers)  
null=lm(Avg.Annual~1,data=pos3BR.removed.outliers)  
error.model.pos3BR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","p3B.removed.outliers",pos3BR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "rbi"   
## [4] "g" "ab" "ageUnder25"   
## [7] "bb"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3572980 -1181896 -414393 1037212 7540564   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -240453919 27559900 -8.725 3.02e-16 \*\*\*  
## hasFreeAgentStatusTRUE 2297128 246462 9.320 < 2e-16 \*\*\*  
## Year 120735 13755 8.778 < 2e-16 \*\*\*  
## rbi 25699 7856 3.271 0.00121 \*\*   
## g -36860 8424 -4.376 1.74e-05 \*\*\*  
## ab 6004 2373 2.530 0.01198 \*   
## ageUnder25TRUE -920246 484687 -1.899 0.05870 .   
## bb 13461 7160 1.880 0.06121 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1894000 on 265 degrees of freedom  
## Multiple R-squared: 0.4718, Adjusted R-squared: 0.4578   
## F-statistic: 33.81 on 7 and 265 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 273 samples  
## 7 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 247, 246, 245, 245, 245, 247, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 1887347 0.4705549 3512.114 1479143 -23.67515 206.6349  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.pos3BR.removed.outliers$pred.err.percent.p3B.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.817 33.020 65.310 198.400 220.300 3261.000

# Random forest for 3b

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 3931835 0.5274244 -31625.66 2578304 -222.6636 244.5524

# 9 3551027 0.5919614 -128913.57 2283713 -151.8158 173.8087

# 16 3544142 0.5905724 -103467.99 2269843 -134.4801 158.3332

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(p3BRfmodelError$pred.err.percent.p3BRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.2143 12.5900 27.9300 61.9700 70.9900 593.4000

p3BRfmodelError=buildAndRunRfModel("Avg.Annual","p3BRfmodel",pos3BR,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "rbi"   
## [4] "hr" "sh" "woba"   
## [7] "double" "h" "a"   
## [10] "g" "r" "po"   
## [13] "sb" "single" "ab"   
## [16] "bb"   
## Random Forest   
##   
## 301 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 240, 242, 239, 241, 242   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 4134944 0.4927900 -34134.35 2663401 -233.2936 255.1234  
## 9 3681067 0.5747236 -60118.07 2279577 -149.0775 170.5657  
## 16 3628482 0.5758457 -77577.59 2249951 -126.2279 149.6329  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(p3BRfmodelError$pred.err.percent.p3BRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2052 12.8600 27.3600 60.8500 72.2900 545.8000

# rANDOM FOREST removed outliers for 3b

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 1953076 0.4827832 -24152.91 1507740 -172.2417 196.7863

# 9 1796763 0.5176729 -25354.37 1325696 -114.3563 138.2963

# 16 1827149 0.4965451 -28362.59 1327702 -103.0686 128.5586

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(error.rf.model.pos3BR.removed.outliers$pred.err.percent.p3B.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.2278 12.7500 27.4900 55.9400 67.7300 429.6000

error.rf.model.pos3BR.removed.outliers=buildAndRunRfModel("Avg.Annual","p3B.rf.removed.outliers",pos3BR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "ab"   
## [4] "age30to35" "g" "hr"   
## [7] "rbi" "h" "a"   
## [10] "ageUnder25" "bb" "age25to30"   
## [13] "r" "age35to50" "ibb"   
## [16] "isAllStar"   
## Random Forest   
##   
## 273 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 220, 219, 218, 218, 217   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 1964653 0.4655221 -28113.76 1503006 -159.1499 183.3796  
## 9 1889623 0.4666303 -66267.64 1394977 -114.8559 139.3446  
## 16 1933437 0.4472321 -59800.18 1400316 -102.1896 128.3452  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(error.rf.model.pos3BR.removed.outliers$pred.err.percent.p3B.rf.removed.outliers)

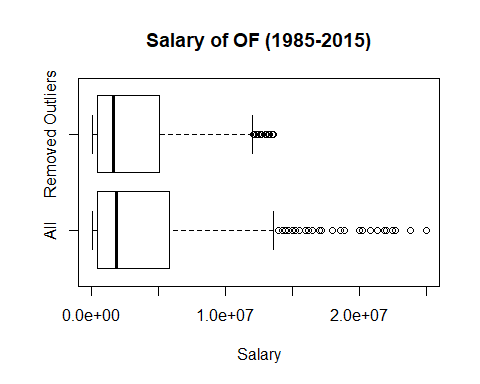
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0479 13.6700 27.9400 54.8200 69.4800 428.7000

# setup data for OF players

regressionFeatures=c("Avg.Annual","Year","woba","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
posOFR=winningBatters[which(winningBatters$pos=="OF"),]  
posOFR=posOFR[,regressionFeatures]

# Show boxplot of salaries and determine if there are outliers

posOFR.removed.outliers=removeOutliers(posOFR,"Avg.Annual")  
  
boxplot(list(posOFR$Avg.Annual,posOFR.removed.outliers$Avg.Annual),xlab="Salary",names=c("All","Removed Outliers"),main="Salary of OF (1985-2015)",horizontal=TRUE)



# Normal model for OF

# RMSE Rsquared ME MAE MPE MAPE

# 3315255 0.5099765 -597.1713 2450935 20.93352 315.7744

# Tuning parameter 'intercept' was held constant at a value of TRUE

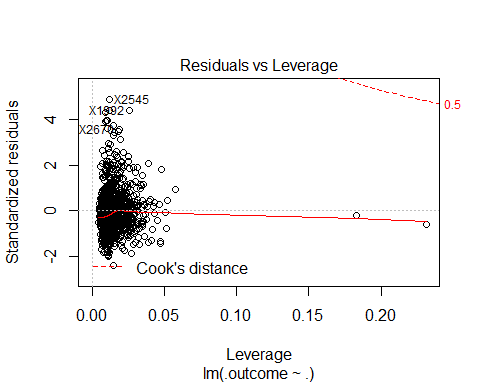
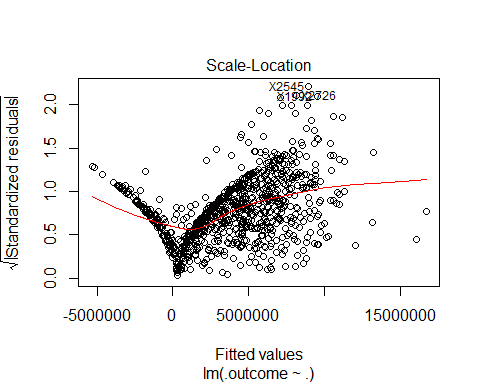
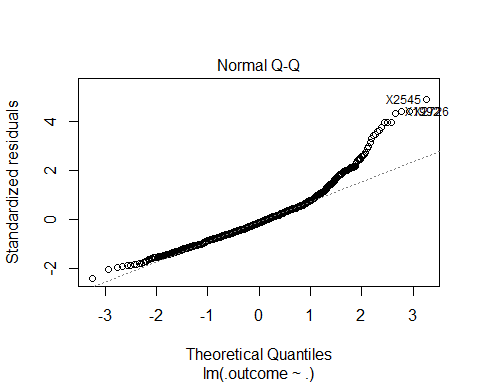
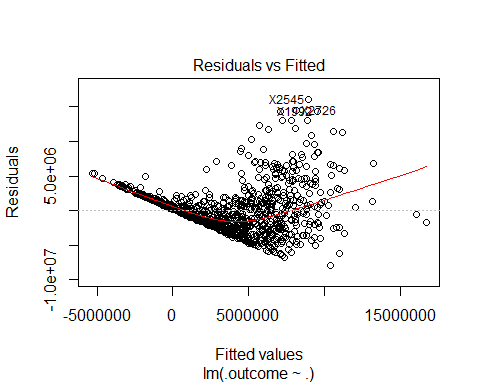
# > summary(pOFModel1Error$pred.err.percent.pOFModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.226 30.480 81.020 310.800 330.000 7258.000

full=lm(Avg.Annual~.,data=posOFR)  
null=lm(Avg.Annual~1,data=posOFR)  
pOFModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","pOFModel1",posOFR,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "ibb"   
## [4] "g" "ab" "triple"   
## [7] "isAllStar" "age30to35" "bb"   
## [10] "sh" "ageUnder25" "g\_idp"   
## [13] "r"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7864385 -2136881 -434541 1476309 16032023   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -439772794 28461476 -15.452 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 2775984 280340 9.902 < 2e-16 \*\*\*  
## Year 220728 14214 15.529 < 2e-16 \*\*\*  
## ibb 134027 28843 4.647 3.90e-06 \*\*\*  
## g -54140 8165 -6.631 5.88e-11 \*\*\*  
## ab 9398 2744 3.425 0.000643 \*\*\*  
## triple -168054 51249 -3.279 0.001083 \*\*   
## isAllStarTRUE 1141666 377006 3.028 0.002533 \*\*   
## age30to35TRUE 783903 264769 2.961 0.003153 \*\*   
## bb 14135 8383 1.686 0.092124 .   
## sh -147024 50898 -2.889 0.003967 \*\*   
## ageUnder25TRUE -855408 465828 -1.836 0.066654 .   
## g\_idp 64472 28894 2.231 0.025916 \*   
## r 23822 13030 1.828 0.067858 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3306000 on 863 degrees of freedom  
## Multiple R-squared: 0.5166, Adjusted R-squared: 0.5094   
## F-statistic: 70.96 on 13 and 863 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 877 samples  
## 13 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 790, 789, 789, 789, 790, 789, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 3303268 0.5103649 9419.623 2443761 22.56472 315.9362  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(pOFModel1Error$pred.err.percent.pOFModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.226 30.480 81.020 310.800 330.000 7258.000

# With salary outliers removed for OF

# RMSE Rsquared ME MAE MPE MAPE

# 2525940 0.5089149 1174.518 1929851 -8.886793 254.453

# Tuning parameter 'intercept' was held constant at a value of TRUE

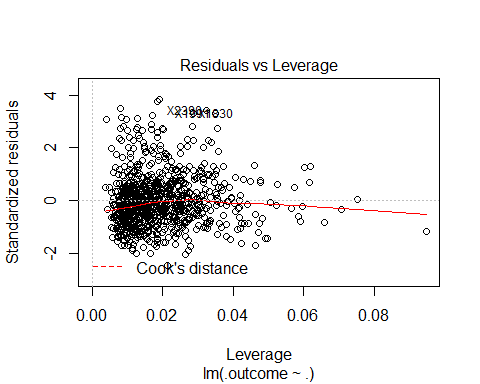
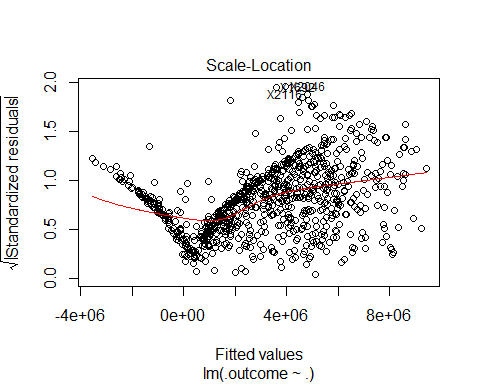
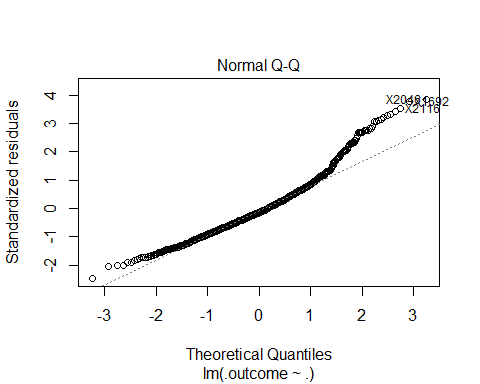
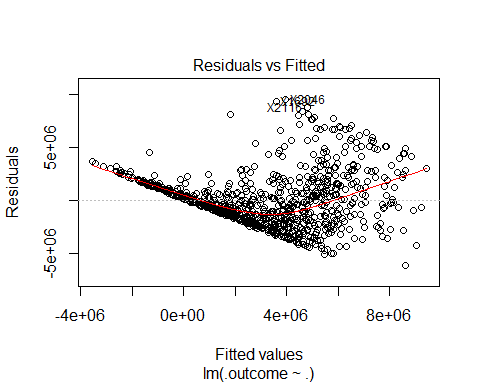
# > summary(error.model.posOFR.removed.outliers$pred.err.percent.pOF.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.077 31.110 71.900 249.000 292.600 5369.000

full=lm(Avg.Annual~.,data=posOFR.removed.outliers)  
null=lm(Avg.Annual~1,data=posOFR.removed.outliers)  
error.model.posOFR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","pOF.removed.outliers",posOFR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "rbi"   
## [4] "g" "bb" "ab"   
## [7] "triple" "isAllStar" "so"   
## [10] "ageUnder25" "sh" "po"   
## [13] "ibb" "cs" "age25to30"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -6117299 -1654007 -369502 1259687 9546957   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -312741830 24434236 -12.799 < 2e-16 \*\*\*  
## hasFreeAgentStatusTRUE 2132809 265839 8.023 3.57e-15 \*\*\*  
## Year 157348 12204 12.893 < 2e-16 \*\*\*  
## rbi 10515 7087 1.484 0.138283   
## g -47651 6314 -7.547 1.19e-13 \*\*\*  
## bb 24074 5874 4.099 4.57e-05 \*\*\*  
## ab 11461 1977 5.796 9.67e-09 \*\*\*  
## triple -144440 39647 -3.643 0.000286 \*\*\*  
## isAllStarTRUE 996762 301186 3.309 0.000976 \*\*\*  
## so -10594 4146 -2.555 0.010790 \*   
## ageUnder25TRUE -1240488 430119 -2.884 0.004029 \*\*   
## sh -89873 43417 -2.070 0.038768 \*   
## po 3903 1543 2.530 0.011605 \*   
## ibb 62281 30467 2.044 0.041251 \*   
## cs -53299 31845 -1.674 0.094571 .   
## age25to30TRUE -403288 260045 -1.551 0.121327   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2512000 on 818 degrees of freedom  
## Multiple R-squared: 0.5194, Adjusted R-squared: 0.5105   
## F-statistic: 58.92 on 15 and 818 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 834 samples  
## 15 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 750, 751, 750, 752, 751, 750, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 2527509 0.5045489 -556.8255 1925006 -11.56441 252.6676  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.posOFR.removed.outliers$pred.err.percent.pOF.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.077 31.110 71.900 249.000 292.600 5369.000

# Random forest for OF

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 3355402 0.5298197 -43128.05 2334165 -204.0826 225.6812

# 9 3154323 0.5576760 -20919.64 2109599 -131.2266 155.6742

# 16 3195145 0.5442133 -17051.10 2115932 -124.1875 150.3024

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(pOFRfmodelError$pred.err.percent.pOFRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.12 11.97 25.97 64.53 76.05 1627.00

pOFRfmodelError=buildAndRunRfModel("Avg.Annual","pOFRfmodel",posOFR,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "woba"   
## [4] "rbi" "g" "isAllStar"   
## [7] "age30to35" "hr" "po"   
## [10] "so" "sh" "ab"   
## [13] "h" "bb" "g\_idp"   
## [16] "sf"   
## Random Forest   
##   
## 877 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 702, 702, 701, 702, 701   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 3290757 0.5523956 -66326.68 2314067 -199.2801 220.7479  
## 9 3058105 0.5841300 -59145.75 2061063 -130.9759 154.6463  
## 16 3108179 0.5677095 -36828.43 2063676 -124.8566 150.4990  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(pOFRfmodelError$pred.err.percent.pOFRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0004 10.5300 26.7800 62.6800 73.7100 1692.0000

# rANDOM FOREST removed outliers for OF

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 2599785 0.5092330 -41602.95 1919354 -193.4959 216.5522

# 9 2445015 0.5386406 -55423.03 1723744 -133.7932 157.9723

# 16 2491364 0.5189231 -52435.17 1738016 -127.7762 153.4309

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 9.

# > summary(error.rf.model.posOFR.removed.outliers$pred.err.percent.pOF.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0427 10.8300 26.8000 63.6200 73.3900 1649.0000

error.rf.model.posOFR.removed.outliers=buildAndRunRfModel("Avg.Annual","pOF.rf.removed.outliers",posOFR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "po"   
## [4] "woba" "double" "g"   
## [7] "rbi" "hr" "g\_idp"   
## [10] "ab" "age30to35" "bb"   
## [13] "r" "isAllStar" "so"   
## [16] "single"   
## Random Forest   
##   
## 834 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 666, 667, 666, 670, 667   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 2600471 0.5095160 -20009.07 1919197 -184.2599 207.6063  
## 9 2449994 0.5384951 -15373.60 1720980 -126.9676 151.5687  
## 16 2460065 0.5327564 -16622.24 1707724 -120.8062 146.4508  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 9.

summary(error.rf.model.posOFR.removed.outliers$pred.err.percent.pOF.rf.removed.outliers)

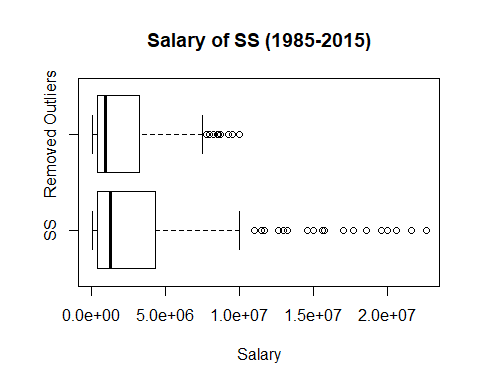
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.005 10.860 26.570 63.050 74.740 1933.000

# setup data for SS players

regressionFeatures=c("Avg.Annual","Year","woba","po","a","e","g","ab","r","h","double","triple","hr","rbi","sb","cs","bb","so","ibb","hbp","sh","sf","g\_idp","isAllStar","isAwardWinner","hasFreeAgentStatus","ageUnder25","age25to30","age30to35","age35to50","single")  
posSSR=winningBatters[which(winningBatters$pos=="SS"),]  
posSSR=posSSR[,regressionFeatures]

# Show boxplot of salaries and determine if there are outliers

posSSR.removed.outliers=removeOutliers(posSSR,"Avg.Annual")  
boxplot(list(posSSR$Avg.Annual,posSSR.removed.outliers$Avg.Annual),xlab="Salary",names=c("SS","Removed Outliers"),main="Salary of SS (1985-2015)",horizontal=TRUE)



# Normal model for SS

# RMSE Rsquared ME MAE MPE MAPE

# 3030541 0.5458726 -15057.02 2297434 25.24792 327.4557

# Tuning parameter 'intercept' was held constant at a value of TRUE

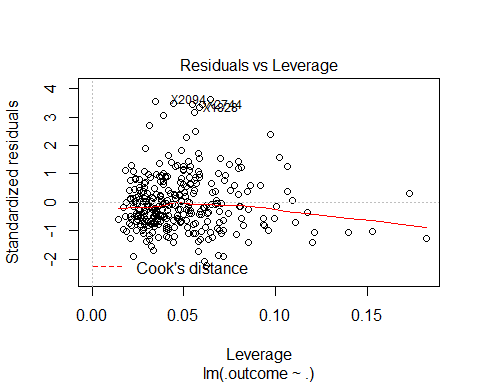
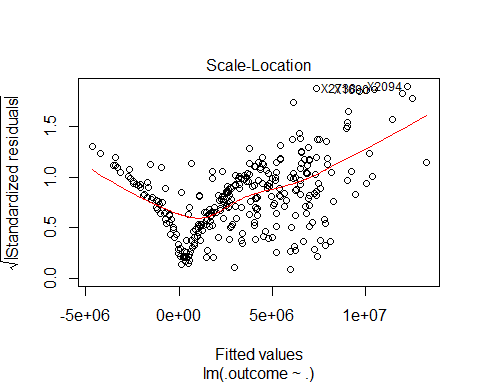
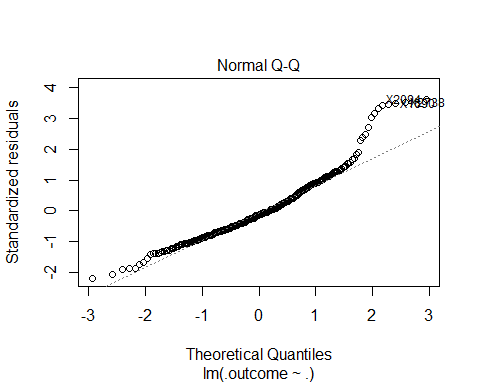
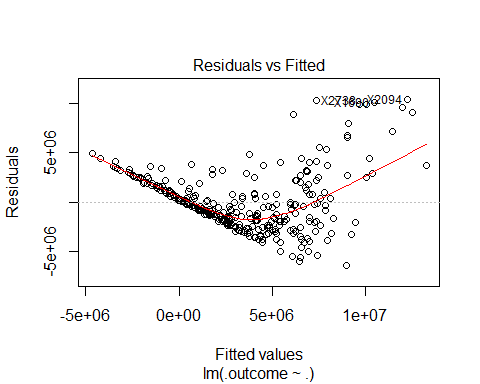
# > summary(pSSModel1Error$pred.err.percent.pSSModel1)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.40 38.98 99.64 311.90 299.30 5095.00

full=lm(Avg.Annual~.,data=posSSR)  
null=lm(Avg.Annual~1,data=posSSR)  
pSSModel1Error=buildAndRunLinearModel(full,null,"Avg.Annual","pSSModel1",posSSR,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "woba"   
## [4] "isAllStar" "g\_idp" "sf"   
## [7] "hbp" "g" "ab"   
## [10] "triple" "rbi" "ageUnder25"   
## [13] "ibb" "r"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -6357422 -1958158 -407313 1525221 10394579   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -331154335 44789034 -7.394 1.52e-12 \*\*\*  
## hasFreeAgentStatusTRUE 3671700 377689 9.721 < 2e-16 \*\*\*  
## Year 163913 22327 7.341 2.11e-12 \*\*\*  
## woba 15827661 6555653 2.414 0.01638 \*   
## isAllStarTRUE 1312444 555431 2.363 0.01879 \*   
## g\_idp 136843 47714 2.868 0.00443 \*\*   
## sf -129594 85919 -1.508 0.13255   
## hbp 95691 53569 1.786 0.07509 .   
## g -67442 15516 -4.347 1.91e-05 \*\*\*  
## ab 14494 5008 2.894 0.00409 \*\*   
## triple -148564 76112 -1.952 0.05190 .   
## rbi -34733 16140 -2.152 0.03221 \*   
## ageUnder25TRUE -872729 587764 -1.485 0.13867   
## ibb 131867 71229 1.851 0.06513 .   
## r 32074 19953 1.607 0.10903   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2973000 on 292 degrees of freedom  
## Multiple R-squared: 0.5795, Adjusted R-squared: 0.5593   
## F-statistic: 28.74 on 14 and 292 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 307 samples  
## 14 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 277, 276, 275, 276, 278, 275, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 3020012 0.5452317 -2939.119 2301633 26.37658 325.264  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(pSSModel1Error$pred.err.percent.pSSModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.40 38.98 99.64 311.90 299.30 5095.00

# With salary outliers removed for SS

# RMSE Rsquared ME MAE MPE MAPE

# 1749254 0.5310168 4443.253 1367256 -10.55374 205.2732

# Tuning parameter 'intercept' was held constant at a value of TRUE

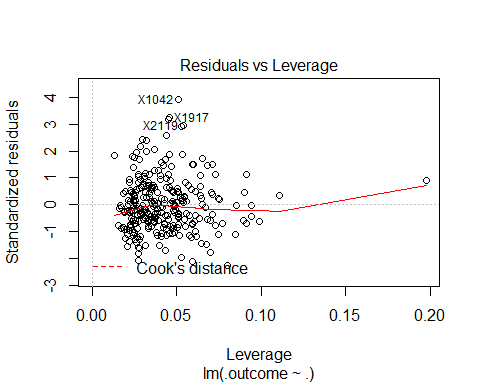
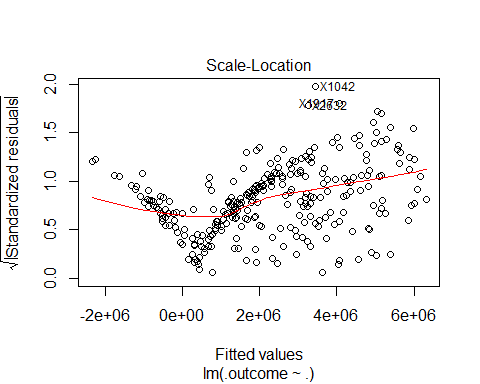
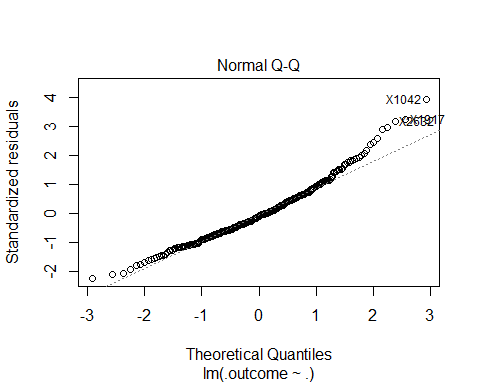
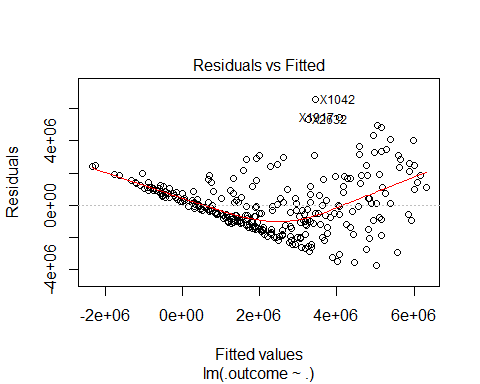
# > summary(error.model.posSSR.removed.outliers$pred.err.percent.pSS.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.144 28.460 75.280 192.800 244.500 3487.000

full=lm(Avg.Annual~.,data=posSSR.removed.outliers)  
null=lm(Avg.Annual~1,data=posSSR.removed.outliers)  
error.model.posSSR.removed.outliers=buildAndRunLinearModel(full,null,"Avg.Annual","pSS.removed.outliers",posSSR.removed.outliers,modelCtrlLM)

## [1] "hasFreeAgentStatus" "Year" "g"   
## [4] "sh" "ab" "ageUnder25"   
## [7] "e" "age30to35" "isAwardWinner"   
## [10] "g\_idp" "bb"   
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3714501 -1131234 -182094 933320 6558280   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -198425929 28622108 -6.933 3.00e-11 \*\*\*  
## hasFreeAgentStatusTRUE 2211455 249702 8.856 < 2e-16 \*\*\*  
## Year 99613 14270 6.980 2.25e-11 \*\*\*  
## g -39716 8654 -4.589 6.80e-06 \*\*\*  
## sh -67422 23541 -2.864 0.00451 \*\*   
## ab 9380 2175 4.313 2.26e-05 \*\*\*  
## ageUnder25TRUE -704313 337373 -2.088 0.03776 \*   
## e 45702 20336 2.247 0.02542 \*   
## age30to35TRUE 382204 260629 1.466 0.14368   
## isAwardWinnerTRUE 548421 337598 1.624 0.10543   
## g\_idp 44153 26855 1.644 0.10130   
## bb 13174 8128 1.621 0.10620   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1718000 on 272 degrees of freedom  
## Multiple R-squared: 0.5664, Adjusted R-squared: 0.5488   
## F-statistic: 32.3 on 11 and 272 DF, p-value: < 2.2e-16  
##   
## Linear Regression   
##   
## 284 samples  
## 11 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 256, 256, 256, 256, 255, 256, ...   
## Resampling results:  
##   
## RMSE Rsquared ME MAE MPE MAPE   
## 1749106 0.5501048 -10660.65 1373586 -19.84008 204.4994  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE



## NULL

summary(error.model.posSSR.removed.outliers$pred.err.percent.pSS.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.144 28.460 75.280 192.800 244.500 3487.000

# Random forest for SS

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 2958802 0.6255291 -4453.171 1965279 -204.3876 224.6459

# 9 2526136 0.6946028 -88365.920 1628820 -136.6045 157.8509

# 16 2495107 0.6952537 -88649.915 1581831 -122.1300 145.4812

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(pSSRfmodelError$pred.err.percent.pSSRfmodel)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.0544 9.4400 21.9600 55.1800 59.9900 559.1000

pSSRfmodelError=buildAndRunRfModel("Avg.Annual","pSSRfmodel",posSSR,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "hr"   
## [4] "double" "rbi" "sb"   
## [7] "a" "single" "h"   
## [10] "ab" "r" "g\_idp"   
## [13] "woba" "hbp" "age35to50"   
## [16] "g"   
## Random Forest   
##   
## 307 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 246, 245, 247, 245, 245   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 2989831 0.6156002 -31372.09 2023949 -215.1269 236.3096  
## 9 2583152 0.6830389 -46285.30 1657103 -136.9670 159.1563  
## 16 2549383 0.6828647 -19665.20 1609179 -122.8298 146.4728  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(pSSRfmodelError$pred.err.percent.pSSRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1035 9.5420 24.0000 59.0900 63.0000 564.0000

# rANDOM FOREST removed outliers for SS

# mtry RMSE Rsquared ME MAE MPE MAPE

# 2 1908341 0.4836230 -24360.24 1380747 -174.5521 197.5533

# 9 1717782 0.5488542 -77887.60 1211558 -132.2739 155.1663

# 16 1689733 0.5579524 -56759.86 1166022 -117.6027 142.3188

# RMSE was used to select the optimal model using the smallest value.

# The final value used for the model was mtry = 16.

# > summary(error.rf.model.posSSR.removed.outliers$pred.err.percent.pSS.rf.removed.outliers)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 0.104 8.804 25.500 57.690 62.090 501.800

error.rf.model.posSSR.removed.outliers=buildAndRunRfModel("Avg.Annual","pSS.rf.removed.outliers",posSSR.removed.outliers,modelCtrlRF)

## [1] "hasFreeAgentStatus" "Year" "hr"   
## [4] "double" "woba" "rbi"   
## [7] "age30to35" "a" "sh"   
## [10] "isAwardWinner" "po" "ageUnder25"   
## [13] "ab" "g\_idp" "h"   
## [16] "single"   
## Random Forest   
##   
## 284 samples  
## 16 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 227, 228, 226, 228, 227   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared ME MAE MPE MAPE   
## 2 1879539 0.5007572 -26649.53 1381604 -171.2569 194.9629  
## 9 1713487 0.5712207 -46773.01 1196083 -123.7410 147.5175  
## 16 1694420 0.5840388 -30986.33 1157351 -113.0670 138.7653  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 16.

summary(error.rf.model.posSSR.removed.outliers$pred.err.percent.pSS.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0348 10.1800 25.2300 58.6800 57.9800 578.3000

# Regression results for base model

# Base model average percent error is about 205%

summary(basicModelError$pred.err.percent.basicModel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.066 35.720 95.430 319.500 357.600 7874.000

summary(remove.outliers.basicModelError$pred.err.percent.basicModel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.069 34.460 72.260 205.600 263.200 3989.000

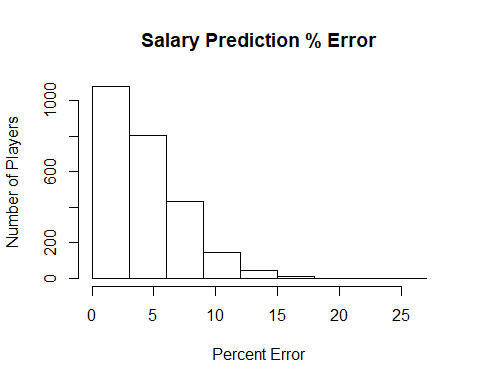
# Log Model with salary outliers

# Avg Error Percent is 4.2%

summary(pAllLogModelError$pred.err.percent.pAllLogModel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.001651 1.739000 3.570000 4.240000 6.007000 22.360000

hist(error.logmodel.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers,breaks=c(0,3,6,9,12,15,18,21,24,27),xlab="Percent Error",ylab="Number of Players",main="Salary Prediction % Error")



summary(pAllLogModelError$pred.err.percent.inverse.logpAllLogModel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0264 23.9200 47.7900 71.5300 75.4400 1238.0000

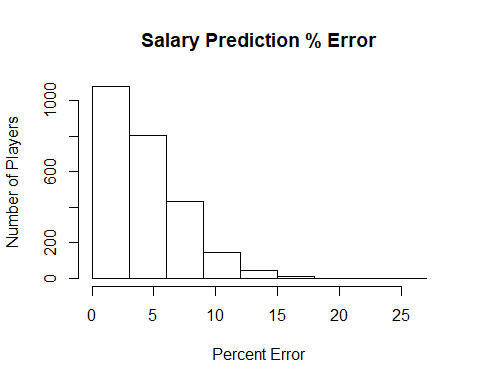
# Log Model with salary outliers removed

# Avg Error Percent is 4.2%

summary(error.logmodel.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.001393 1.715000 3.539000 4.207000 6.037000 21.700000

hist(error.logmodel.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers,breaks=c(0,3,6,9,12,15,18,21,24,27),xlab="Percent Error",ylab="Number of Players",main="Salary Prediction % Error")



summary(error.logmodel.posAllR.removed.outliers$pred.err.percent.inverse.logpAll.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.018 23.670 46.750 68.700 75.140 1139.000

# Show Regression Results for All players

# Random forest produced the best model with mean percent error of 50%

summary(pAllModel1Error$pred.err.percent.pAllModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.003 34.440 94.420 313.800 323.700 8208.000

summary(error.model.posAllR.removed.outliers$pred.err.percent.pAll.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.034 32.550 74.400 197.900 226.500 4854.000

summary(pAllRfmodelError$pred.err.percent.pAllRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0074 11.2300 25.5400 57.0300 68.9800 1040.0000

summary(error.rf.model.posAllR.removed.outliers$pred.err.percent.pAll.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.02 10.91 23.85 51.12 63.69 961.80

# Summary of regression for C players

# random forest produced the best model with 45% mean error

summary(pCModel1Error$pred.err.percent.pCModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.455 34.490 109.900 259.400 266.800 5496.000

summary(error.model.posCR.removed.outliers$pred.err.percent.pC.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0614 31.2000 62.4000 118.6000 154.5000 1808.0000

summary(pCRfmodelError$pred.err.percent.pCRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.097 12.900 24.380 62.260 70.120 2470.000

summary(error.rf.model.posCR.removed.outliers$pred.err.percent.pC.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0031 10.9400 22.5600 44.1400 51.6200 862.8000

# Summary of regression for 1B players

# random forest produced the best model with 64% mean error

summary(p1BModel1Error$pred.err.percent.p1BModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.47 36.84 97.50 374.40 349.80 9865.00

summary(error.model.pos1BR.removed.outliers$pred.err.percent.1B.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.239 32.790 81.510 241.600 276.400 6159.000

summary(p1BRfmodelError$pred.err.percent.p1BRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1489 10.7500 27.3800 72.4000 84.2700 853.3000

summary(error.rf.model.pos1BR.removed.outliers$pred.err.percent.p1B.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.05 11.20 27.81 64.18 75.89 1124.00

# Summary of regression for 2B players

# random forest produced the best model with 50% mean error

summary(p2BModel1Error$pred.err.percent.p2BModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.184 34.270 81.300 216.900 246.000 3192.000

summary(error.model.pos2BR.removed.outliers$pred.err.percent.p2B.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0962 31.1600 73.3000 161.1000 186.2000 1587.0000

summary(p2BRfmodelError$pred.err.percent.p2BRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2337 10.9000 24.3800 56.5000 68.5800 465.9000

summary(error.rf.model.pos2BR.removed.outliers$pred.err.percent.p2B.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.028 12.500 25.170 51.240 56.550 499.900

# Summary of regression for 3B players

# random forest produced the best model with 55% mean error

summary(p3BModel1Error$pred.err.percent.p3BModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.536 36.230 97.610 367.700 326.800 5989.000

summary(error.model.pos3BR.removed.outliers$pred.err.percent.p3B.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.817 33.020 65.310 198.400 220.300 3261.000

summary(p3BRfmodelError$pred.err.percent.p3BRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.2052 12.8600 27.3600 60.8500 72.2900 545.8000

summary(error.rf.model.pos3BR.removed.outliers$pred.err.percent.p3B.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0479 13.6700 27.9400 54.8200 69.4800 428.7000

# Summary of regression for OF players

# random forest produced the best model with 63% mean error

summary(pOFModel1Error$pred.err.percent.pOFModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.226 30.480 81.020 310.800 330.000 7258.000

summary(error.model.posOFR.removed.outliers$pred.err.percent.pOF.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.077 31.110 71.900 249.000 292.600 5369.000

summary(pOFRfmodelError$pred.err.percent.pOFRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0004 10.5300 26.7800 62.6800 73.7100 1692.0000

summary(error.rf.model.posOFR.removed.outliers$pred.err.percent.pOF.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.005 10.860 26.570 63.050 74.740 1933.000

# Summary of regression for SS players

# random forest produced the best model with 55% mean error

summary(pSSModel1Error$pred.err.percent.pSSModel1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.40 38.98 99.64 311.90 299.30 5095.00

summary(error.model.posSSR.removed.outliers$pred.err.percent.pSS.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.144 28.460 75.280 192.800 244.500 3487.000

summary(pSSRfmodelError$pred.err.percent.pSSRfmodel)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1035 9.5420 24.0000 59.0900 63.0000 564.0000

summary(error.rf.model.posSSR.removed.outliers$pred.err.percent.pSS.rf.removed.outliers)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0348 10.1800 25.2300 58.6800 57.9800 578.3000