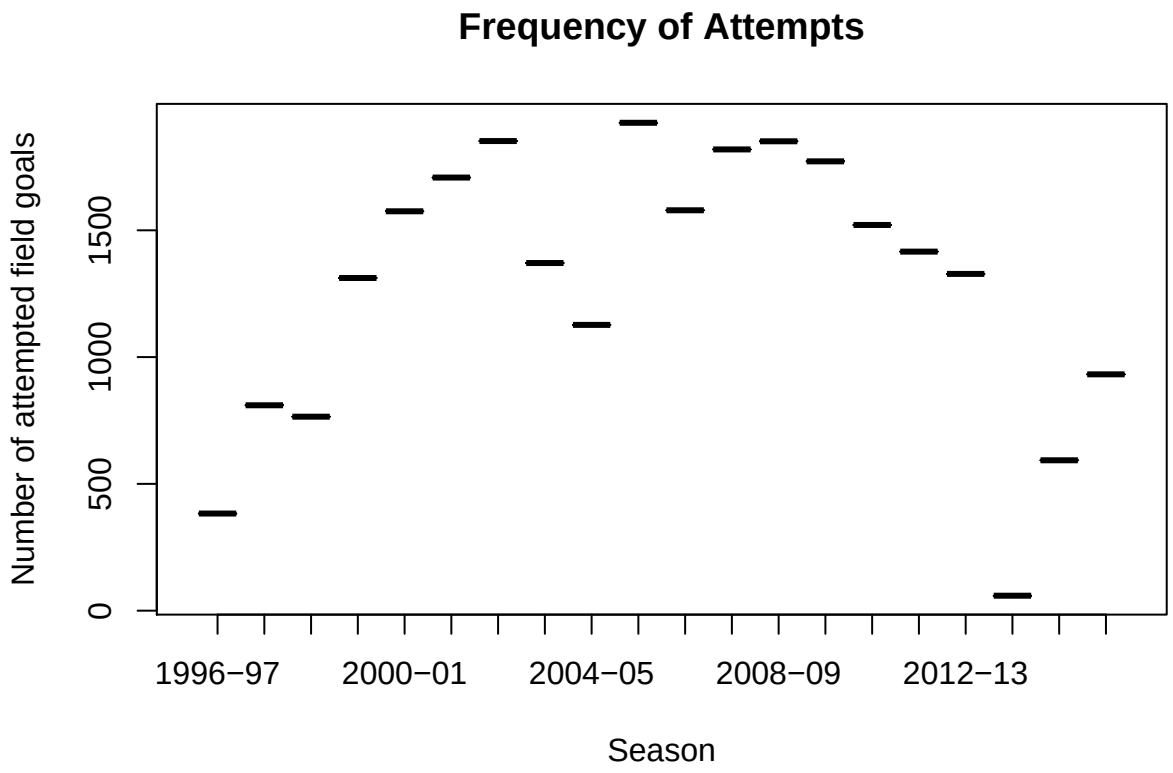
pstat 231

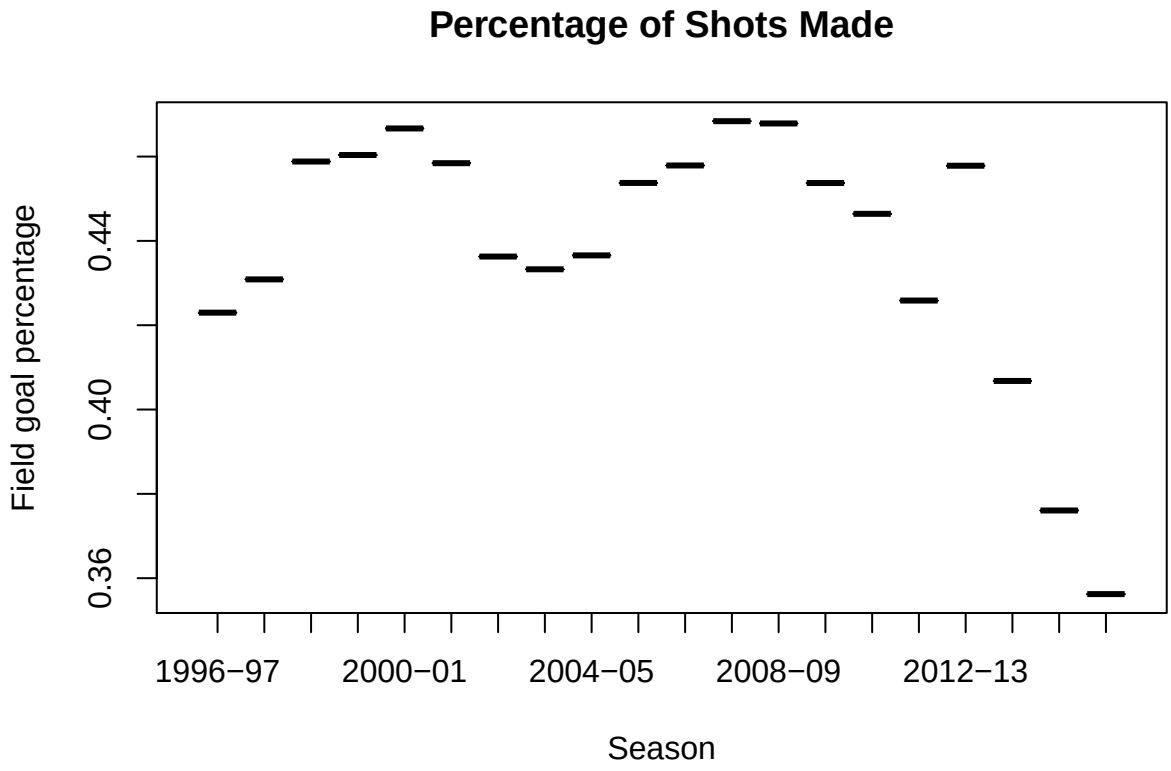
```
setwd("/Users/Shawn/Desktop/PSTAT231-Project")
set.seed(1)
# import data
library(data.table)
kobe = data.table(read.csv("kobe.csv"))
kobe = kobe[complete.cases(kobe),]
# randomly partition data into training set and test set
test.index = sample(seq_len(nrow(kobe)), size = floor(nrow(kobe)*0.1),replace = F)
train.index = setdiff(seq_len(nrow(kobe)),test.index)

shot.season=aggregate(shot_made_flag~season, kobe, length) # number of shots made per season
plot(shot.season,xlab="Season",ylab="Number of attempted field goals", main="Frequency of Attempts")
```




```
season.avg=aggregate(shot_made_flag~season,kobe,mean) # goal percentage per season
plot(season.avg,xlab="Season",ylab="Field goal percentage", main="Percentage of Shots Made")

# transform data to type date
kobe$game_date = as.Date(kobe$game_date, "%Y-%m-%d")
# latitude and longitude are obviously useless
# game_id, game_event_id,team_id,team_name also useless
kobe[,lon:=NULL]
kobe[,lat:=NULL]
kobe[,game_id:=NULL]
kobe[,game_event_id:=NULL]
kobe[,team_id:=NULL]
```

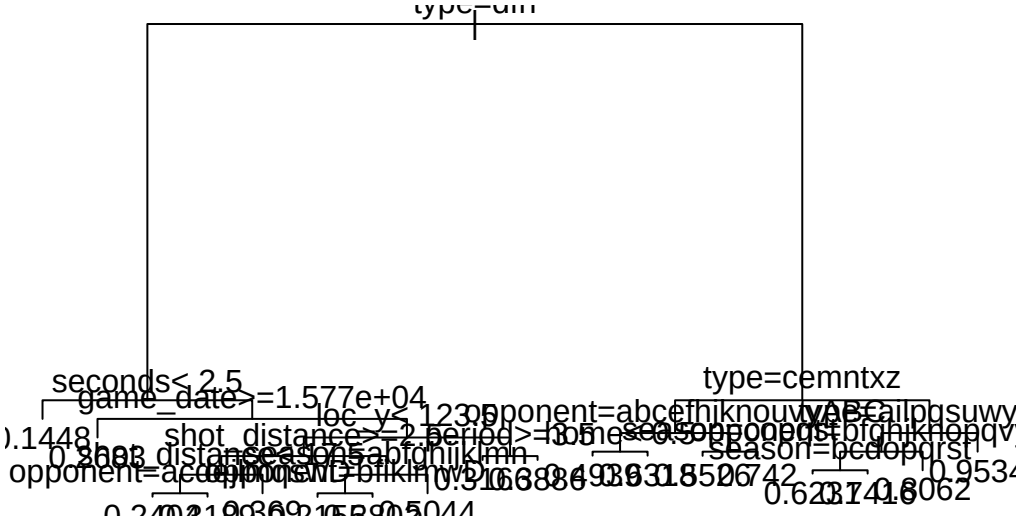
```
kobe[,team_name:=NULL]

# combine minuetes remaining and seconds remaining
kobe[,seconds:=minutes_remaining*60+seconds_remaining,by=1:nrow(kobe)]
kobe[,minutes_remaining:=NULL]
kobe[,seconds_remaining:=NULL]

# replace matchup with binary variable "home"
kobe[,home:=1]
kobe[substr(matchup,5,5)=='@',home:=0]
kobe[,matchup:=NULL]

# some teams changed their names or locations
# we need combine old names and new names
kobe[opponent=="NOH",opponent:="NOP"]
kobe[opponent=="VAN",opponent:="MEM"]
kobe[opponent=="SEA",opponent:="OKC"]
kobe[opponent=="NJN",opponent:="BKN"]
library(rminer)
```

```
## Loading required package: kknn
```




```
kobe$opponent = delevels(kobe$opponent, c("NOH","VAN","SEA","NJN"), label = NULL)
# watch for the correlation between loc_x/loc_y with shot zone/shot distance
# library(corrplot)
# kobe.keep = kobe[,c("period","shot_distance","loc_x","loc_y","playoffs"),with=F]
# corr = corrplot(cor(kobe.keep))
# shot distance has a high correlation between loc_y
```

```r
# kobe[,loc_y:=NULL]
# but it doesn't effect our model very much

# check if we need action_type
action.glm = glm(data = kobe, shot_made_flag~action_type)
# summary(action.glm) output too long -> hide
combine.glm = glm(data = kobe,shot_made_flag~combined_shot_type)
# summary(combine.glm) output too long -> hide

# not all action type are statistically significant
# we will replace action_type with combined_type
# if specific action_type is infrequent compared to others

freqTable = data.table(action_types = levels(kobe$action_type),frequency = as.vector(table(kobe$action_
freqTable =  freqTable[frequency>=50]
#freqTable
kobe[,type:=combined_shot_type,by=1:nrow(kobe)]
kobe[action_type %in% freqTable$action_types,type:=action_type]

# now we have these levels in our new varible "type"
#levels(kobe$type)
# delete action_type and combined_type
kobe[,action_type:=NULL]
kobe[,combined_shot_type:=NULL]
########################## data cleaning complete ################################

# use chi-square test to see if these "distance/location variable" are independent
tbl1 = table(kobe$shot_type,kobe$shot_zone_area)
chi1 = chisq.test(tbl1,correct = F)
#chi1

tbl2 = table(kobe$shot_type,kobe$shot_zone_basic)
chi2 = chisq.test(tbl2,correct = F)
#chi2

tbl3 = table(kobe$shot_type,kobe$shot_zone_area)
chi3 = chisq.test(tbl3,correct = F)
#chi3
# as expected they are highly dependent
# we will eventually drop them in the following procedure

# decision tree
library(rpart)
orig.tree = rpart(data = kobe,formula = shot_made_flag~.-shot_id,na.action = NULL,control=rpart.control
plot(orig.tree)
text(orig.tree)
```

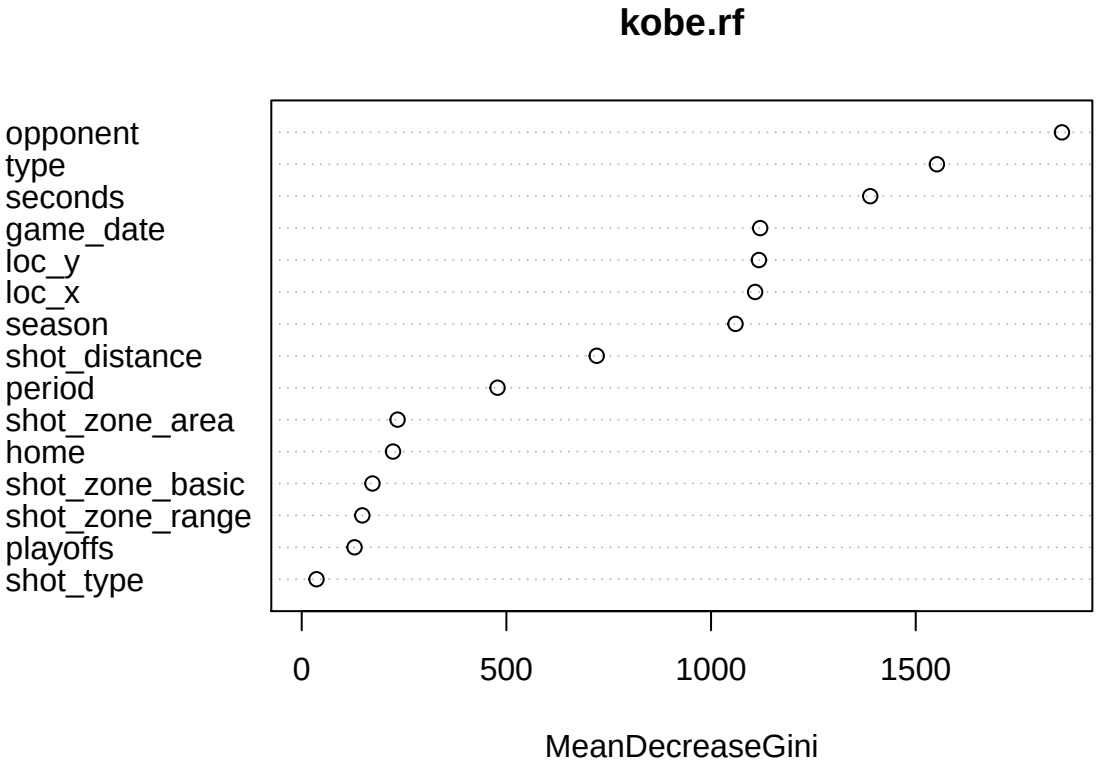



```
# interesting fact: Kobe shot poorly at the last 2.5 seconds

# random forrest
library(randomForest)
```

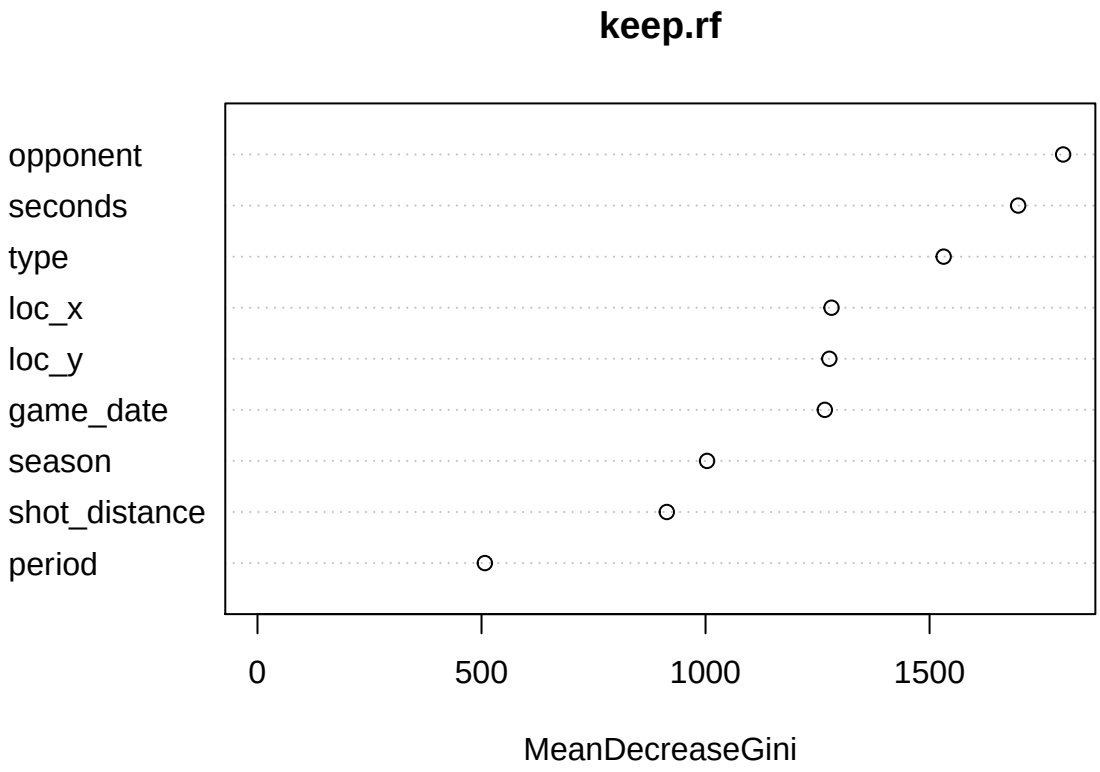
```
## randomForest 4.6-12
```

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

```
kobe.rf = randomForest(formula=as.factor(shot_made_flag)~.-shot_id,na.action=NULL,data = kobe,ntree=500)
imptplot = varImpPlot(kobe.rf)
```



```
imptplot = as.data.table(imptplot,keep.rownames = T)
impt.sort = setorder(imptplot, cols = "MeanDecreaseGini")

# Important variables:
# opponent, type, seconds, loc_x, game_date, season, shot_distance, period
# 1,2,3,5,6,7,12,13,14,15,17  index of variable
col.keep = c(impt.sort$rn[7:15],"shot_made_flag","shot_id")
kobe.keep = subset(kobe,select = col.keep)

# actuall prediction
test = kobe.keep[test.index,]
train = kobe.keep[train.index,]

# redo random forrest on training data
keep.rf = randomForest(formula=as.factor(shot_made_flag)~.-shot_id,na.action=NULL,data = train,ntree=500
imptplot = varImpPlot(keep.rf)
```




```
#imptplot
```

```
# random forrest prediction on training data
rand.train = predict(keep.rf,train,type="class")
rand.train.table = table(rand.train,train$shot_made_flag)
rf.train.error = (rand.train.table[3] + rand.train.table[2])/nrow(train)
rf.train.error
```

```
## [1] 0.0003026634
```

```r
# random forrest prediction on test data
rand.test = predict(keep.rf,test,type="class")
rand.conti.table = table(rand.test,test$shot_made_flag)
rf.error.rate = (rand.conti.table[3] + rand.conti.table[2])/nrow(test)
rf.error.rate
```

```
## [1] 0.3479953
```

```r
glm.fit = glm(data = train, shot_made_flag~.-shot_id,family = binomial)
#glm.fit

# logistic regresssion prediciton on training data
glm.probs.train = predict(glm.fit,train,type="response")
glm.pred.train = rep("Shot fail",nrow(train))
glm.pred.train[glm.probs.train>0.5]="Shot Made"
glm.conti.table.train = table(glm.pred.train,train$shot_made_flag)
glm.error.rate.train = (glm.conti.table.train[3] + glm.conti.table.train[2])/nrow(train)
glm.error.rate.train
```

```
## [1] 0.3196126
```

```r
# logistic regression prediction on testing data
glm.probs = predict(glm.fit,test,type="response")
glm.pred=rep("Shot fail",nrow(test))
glm.pred[glm.probs>0.5]="Shot Made"
glm.conti.table = table(glm.pred,test$shot_made_flag)
glm.error.rate = (glm.conti.table[3] + glm.conti.table[2])/nrow(test)
glm.error.rate
```

```
## [1] 0.3164656
```