How NFL Scouting Combine Results Influence Outcomes of the NFL Draft

Julius Kintop

STAT 4893W

3/30/23

Introduction:

The National Football League (NFL) Draft is an annual event where all 32 NFL teams select eligible college football players to join their rosters. The draft process is a critical aspect of team-building, as it directly impacts the team's future performance and success. General managers and coaches must evaluate and rank players based on their potential to excel in the NFL, making the draft a complex and high-stakes decision-making task. An essential part of this evaluation process is the NFL Scouting Combine, where players participate in various physical tests and drills designed to assess their athleticism, football-related skills, and personality traits via interviews. What we are interested in here is purely the physical portion of the combine. I chose this topic because the 2023 NFL draft is coming up in late April, and the NFL combine is one of the major events that scouts and executives use to rank players.

In recent years, the use of statistical methods and machine learning techniques has gained prominence in sports analytics, with the goal of improving decision-making and performance evaluation. This study aims to explore the predictive power of NFL combine data in determining a player's draft status and draft pick position. By leveraging logistic regression, linear regression and random forest models, we seek to identify the most influential drills in predicting draft success and compare the performance of these methods.

Random forests and decision trees are popular machine learning techniques used for both classification and regression tasks. Both are “Tree Based” methods which make predictions based on a partition of the feature space.

Methods and Materials:

The data that is used in this study was collected from the public domain which is stored in a public database free to download. The two variables of interest here are whether or not a player is drafted (Drafted) and Draft Position (Pick). The features that are used to predict these variables are the various drills performed at the combine meant to measure athletic skills, which are the 40 yard dash (which measures sprint speed), bench press (strength), 3 cone drill (agility), Shuttle drill (quickness, change of direction), vertical jump, and broad jump. Many players do not participate in all of these drills, so methods were implemented to account for this in analysis.

Firstly, the data was split using a validation-set approach, with 80% of observations belonging to the training set and 20% of observations using the validation set. Using the training set, I will first predict whether or not a player will be drafted based on their scores in the combine using binomial logistic regression with the response variable being "Drafted? (Yes/No). This method predicts a binary outcome based on the predicted probability that a player is drafted. In this case, if the method gives a probability greater than 0.5 of being drafted, they are predicted to be drafted, otherwise they are predicted to be undrafted.

After predicting every player's draft status, we will then focus on the players that were predicted to be drafted and make predictions for *where* they will be selected in the draft, ranging from pick 1 to pick 256. To do this, we use linear regression to make numerical predictions.

Even though they did not have a realized draft position, players that were undrafted in reality were assigned a pick value of 400 in the data so that they are distinguishable from drafted players but also to account for the fact that many undrafted players get signed to teams after the draft and have a chance at a roster spot.

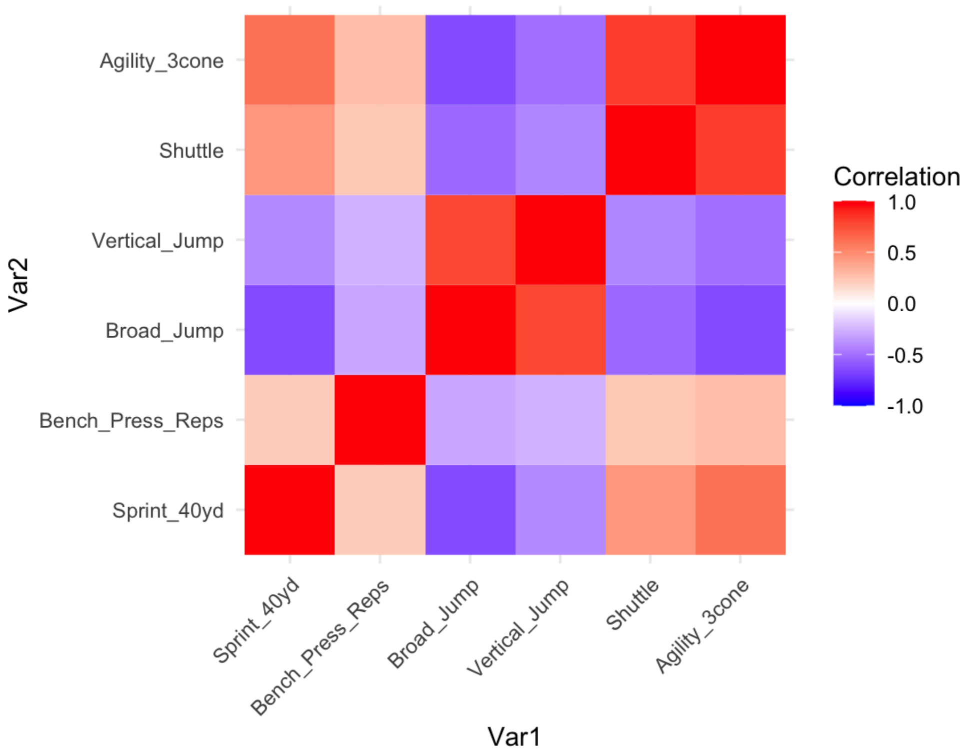
The advanced method used is Random Forest. Random Forest is an ensemble learning method used for both classification and regression tasks. It combines multiple decision trees to create a more robust and accurate model. By averaging the predictions from individual trees, random forests reduce the impact of noise and overfitting, leading to better generalization. Random Forest is built upon decision trees, which are hierarchical models that recursively split the data based on feature values, ultimately reaching a decision at the leaf nodes. Each internal node of the tree represents a decision based on a feature, and each leaf node represents the predicted outcome.

Random Forest uses a technique called bagging to create multiple decision trees. Each tree is trained on a bootstrapped sample (random sample with replacement) of the original dataset. This approach helps reduce overfitting and improves the stability of the model. Feature Randomness: When constructing each decision tree, a random subset of features is considered for splitting at each node. This introduces additional randomness to the model, making it less likely to overfit the training data and allowing it to capture complex relationships between features.

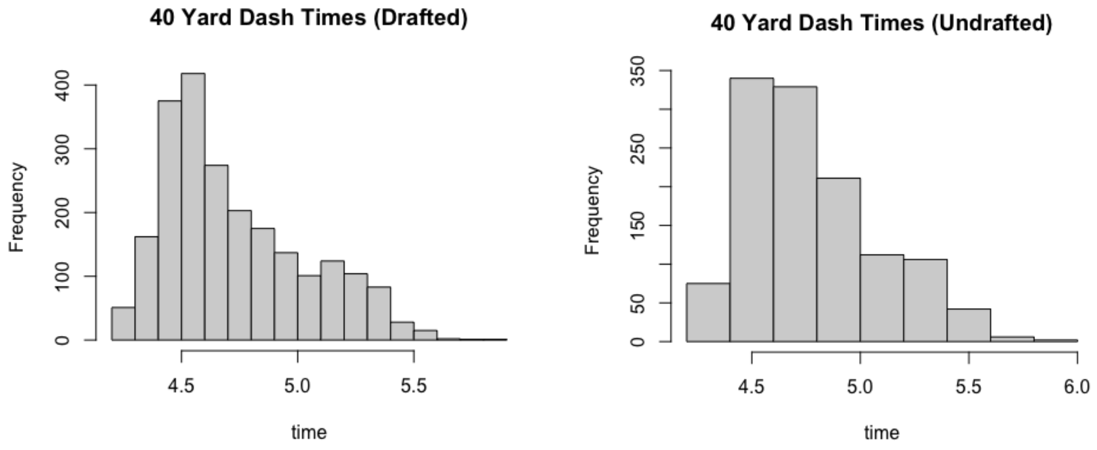
The process is broadly as follows: For each tree (in the random forest), a bootstrapped sample is created by sampling with replacement from the original dataset. Then, at each node, a random subset of m features is considered for splitting, where m << M (typically, m = sqrt(M) for classification and m = M/3 for regression). For classification tasks, the majority vote from all trees determines the predicted class; for regression tasks, the average prediction across all trees is used. Random forest has some assumptions, including that the base learners (decision trees) are weakly correlated. High correlation between trees may reduce the benefits of the ensemble method. The method also assumes that the bootstrapped samples and random feature selection provide enough diversity among the trees to create a robust model. In order to assess the model, we can use Cross-Validation: In this research, a grid search was used to select an optimal random forest model that is robust but not computationally expensive. For classification, the metrics we will use include accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC-ROC). For regression, we will use RMSE, where lower values indicate better performance.

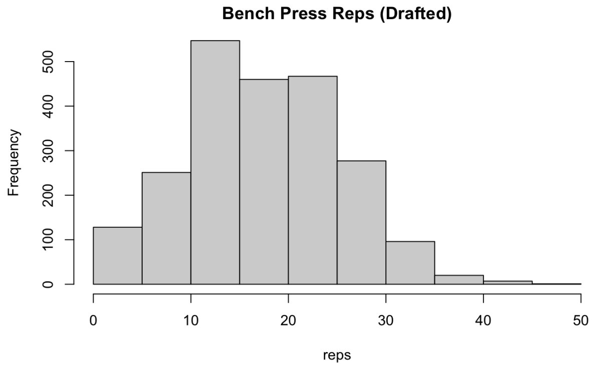
Results

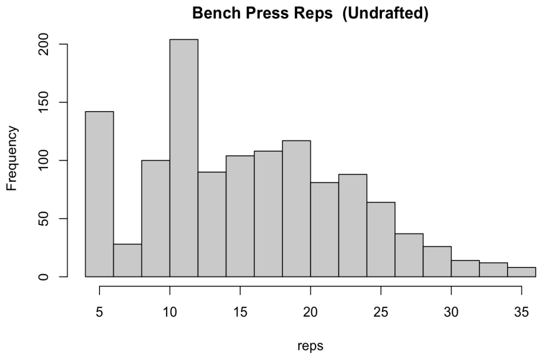
For starters, a few of the drills were correlated with each other, which makes sense considering the more (or less) athletic a player is overall, the better (or worse) they are more likely to perform in any given drill. Players that run are also more likely to jump higher, and some drills measure similar areas of athleticism, for example broad jump and vertical jump both measure a player’s jumping ability, so correlation among these variables is not a surprise.



I wanted to explore the differences in results among drafted and undrafted players, as well as the differences in results among draft pick ranges within the draft. For this, I used the 7 rounds of the draft as a proxy to compare the players within the draft.

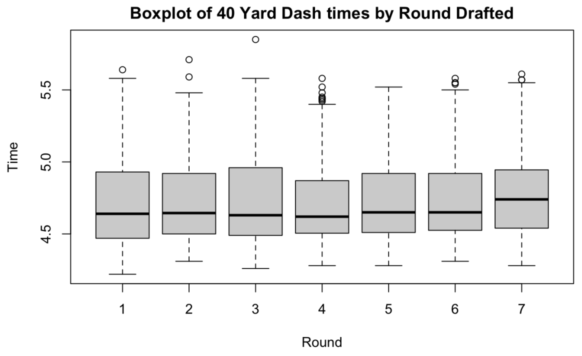


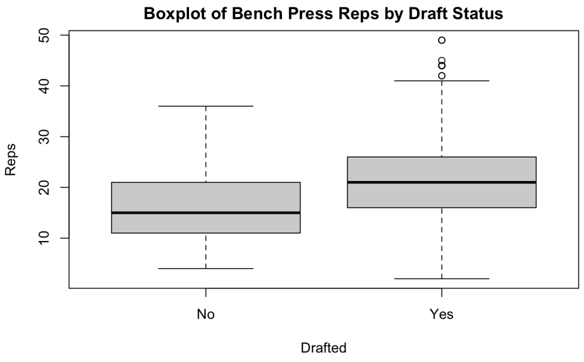


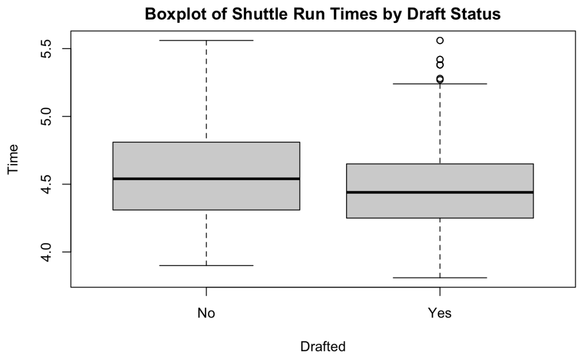


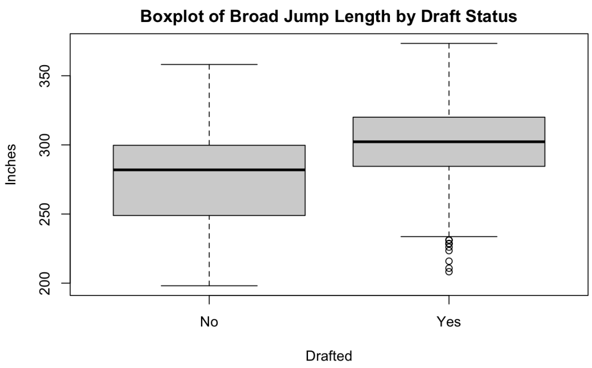
There is a greater skew towards faster sprint speeds among drafted players than undrafted players, which can be a slight indicator that 40 yard dash times matter in the draft process. There is also an interesting anomaly with undrafted players and their bench press, which has a high frequency of low reps on the bench press, which could be an indicator of significance. It was not as easy to decipher any differences among draft ranges however, as their histograms were very similar (see appendix).

Exploring some other variables, there did not seem to be an extreme difference in 40 yard dash times within the draft according to the boxplot, and the same went for the other variables as well.









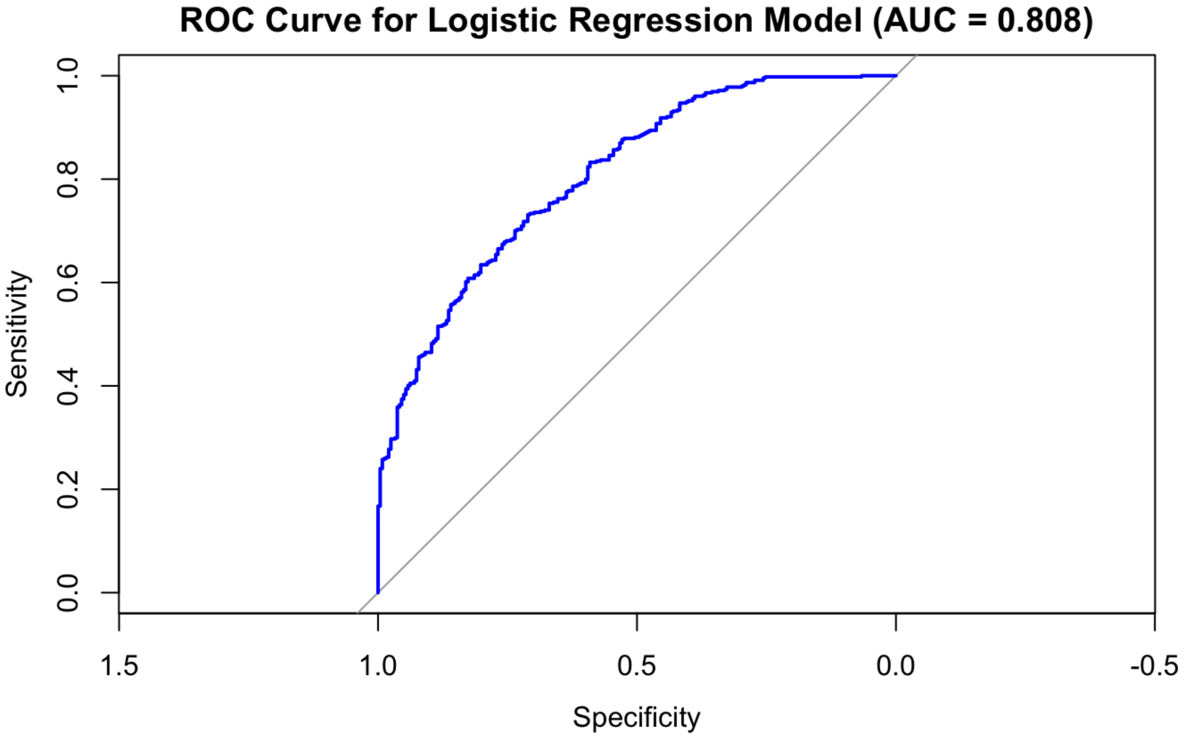
There did seem to be some slight differences in boxplots for bench press, shuttle run, and broad jump between the drafted and undrafted players, which was more robust than the within-draft differences.

The results from the statistical analysis shows us more about what's important here. The Logistic Regression model that is used to predict whether a player will be drafted or not based on their metrics had some very significant outcomes.

Significant Variables in Drafted (Yes/No)

Drill	**P-Value**
40 Yard Dash	0.0242
Broad Jump	2e-16
Shuttle	0.0209
Bench Press	2e-16

These results are extremely significant, which shows us that getting better results in these combine drills is a good predictor of whether or not a player will be drafted in any of the 7 rounds of the draft.



This is the ROC curve, which shows how well our random forest classification model performed in terms of sensitivity and specificity, which are two measures related to the accuracy of the predicted classifications compared to the actual ones.The logistic regression model performed reasonably well all things considered, with an AUC of 0.808, and a total accuracy of 0.7514.

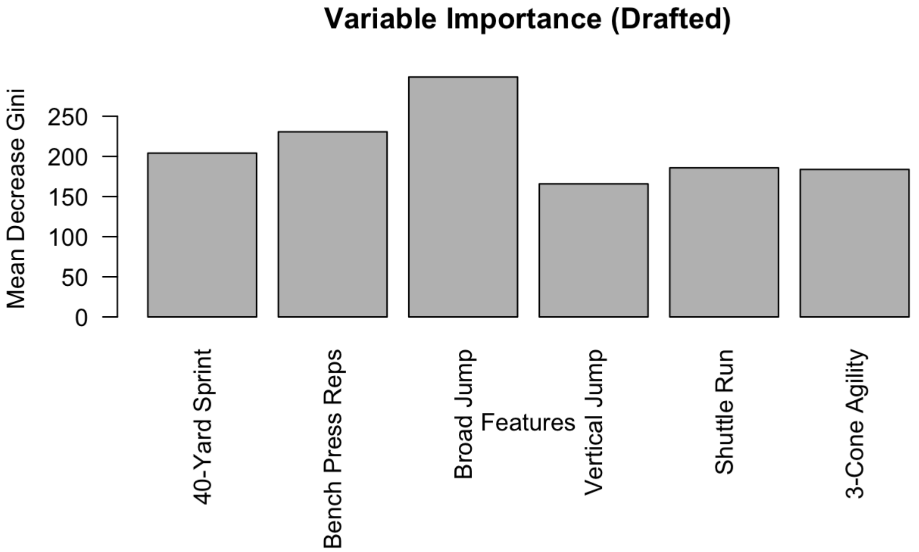
As for the linear regression, which is done to predict *where* players will be drafted (assuming they are predicted to be drafted), there were also significant variables.

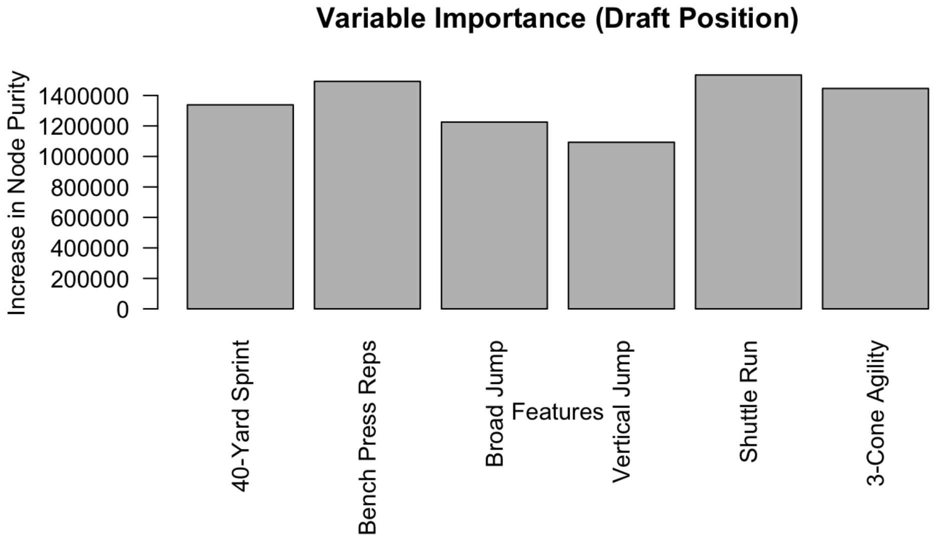
Significant Variables (Draft Position)

Drill	**P-value**
Bench Press	2.67e-10
Broad Jump	1.64e-5
Shuttle Run	0.0203

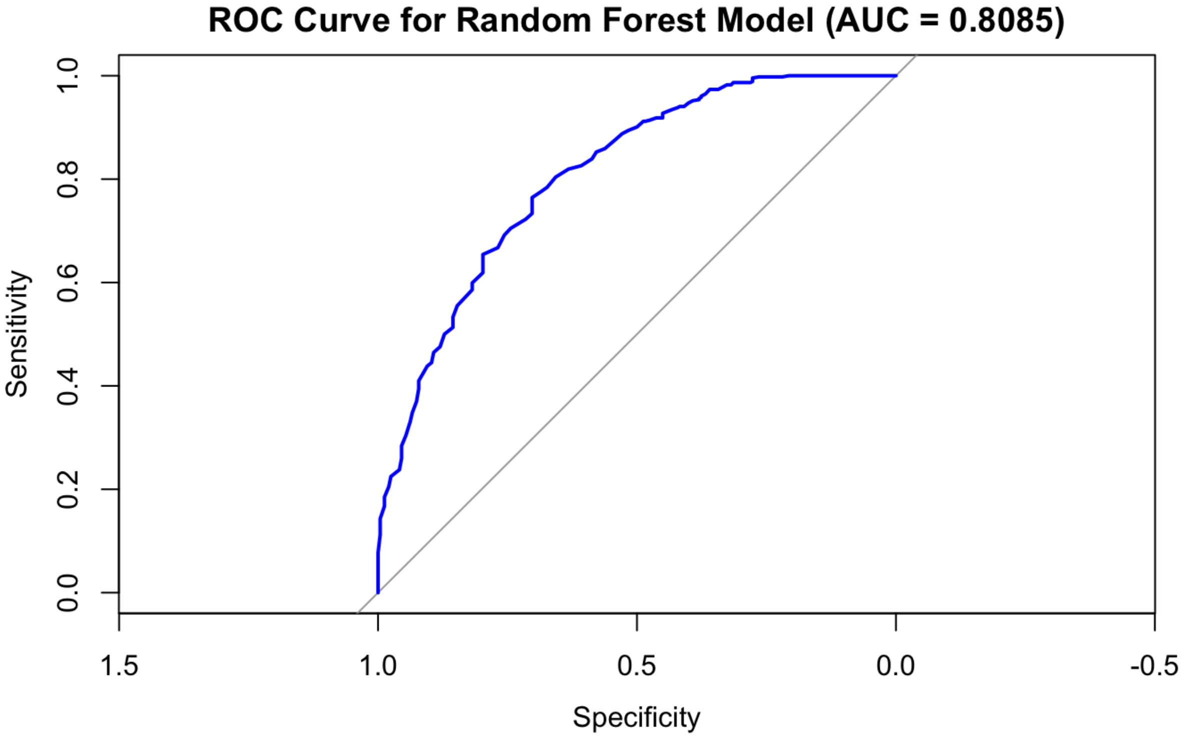
It is important to note that the 40 yard dash was not significant in this model, so it seems that once players cross a certain threshold of speed, the other factors of their athleticism play more of a part in their draft position. The RMSE of the linear regression was 122.22, which shows that the predictions were pretty inaccurate overall, but this can likely be attributed to the fact that undrafted players were given a draft pick value of 400, and the predicted quantity goes up to a maximum of 256, so the error ended up being inflated.

The results of the random forest models were a little bit different. Broad jump was extremely important in predicting draft status, as well as bench press. As for draft position, shuttle run, bench press and 3-cone were all relatively important.





Here, the “Increase in Node Purity” is a measure of how important a feature is in making accurate predictions. The higher the increase in node purity (in this case the “node purity” can be associated with decrease in MSE), 40 yard dash times are considered the most important variable for predicting a player’s draft position. “MeanDecreaseGini”also refers to the feature importance, which is a similar measure to increase in node purity but for classification. The RMSE for the regression random forest is 163.94, which is worse than the linear regression.



The AUC value of 0.8085 is almost the same as the logistic regression.

Discussion and Conclusions

This project is mainly limited by one thing: most players don’t participate in every drill (in fact, it is exceedingly rare that a player does *all* of the drills I used for evaluation). So in order to remedy this, I used iterative regression imputation to predict how players would do in certain drills, given their other performances. The issue with this is that these are not their real measures, so there is inherent added error in the predictions with that alone. But, given the imputed data, the results matched some of the previous intuitions I had about the NFL combine, that scouts value speed more than anything else (for the most part) along with power and quickness. There are many other factors to consider when drafting a player, but their raw athletic testing numbers are a good way to get an idea of how they will match up with their peers on the field.

References

Hastie, T., Tibshirani, R., & Witten, D. (2021). *Introduction to Statistical Learning with Applications in R* (2nd ed.)

Data: Redlineracer. (2019). *NFL Combine Performance Data 2009-2019* [Dataset]. Kaggle.

IBM. (n.d.). *Random forest*. IBM.

Ngari, D. (2020, November 26). *Introduction to random forest in machine learning*. Section Engineering Education.

StataCorp. (n.d.). Hurdle models. Stata.

Appendix

NFL Combine Research Project

Julius Kintop

2023-03-14

Data

```
df = read.csv("NFL.csv", header=TRUE)
head(df)
```

```
##   Year                    Player Age      School Height    Weight Sprint_40yd
## 1 2009    Beanie Wells\\WellCh00  20    Ohio St. 1.8542 106.59421        4.38
## 2 2009      Will Davis\\DaviWi99  22    Illinois 1.8796 118.38761        4.84
## 3 2009  Herman Johnson\\JohnHe23  24         LSU 2.0066 165.10762        5.50
## 4 2009  Rashad Johnson\\JohnRa98  23     Alabama 1.8034  92.07925        4.49
## 5 2009      Cody Brown\\BrowCo96  22 Connecticut 1.8796 110.67654        4.76
## 6 2009 Trevor Canfield\\CanfTr20  23  Cincinnati 1.9304 139.25286        5.28
##   Vertical_Jump Bench_Press_Reps Broad_Jump Agility_3cone Shuttle
## 1         85.09               25     325.12            NA      NA
## 2         83.82               27     292.10          7.38    4.45
## 3            NA               21         NA            NA      NA
## 4         93.98               15     304.80          7.09    4.23
## 5         92.71               26     304.80          7.10    4.40
## 6            NA               29         NA            NA      NA
##                           Drafted..tm.rnd.yr.      BMI Player_Type
## 1  Arizona Cardinals / 1st / 31st pick / 2009 31.00419     offense
## 2 Arizona Cardinals / 6th / 204th pick / 2009 33.51007     defense
## 3 Arizona Cardinals / 5th / 167th pick / 2009 41.00582     offense
## 4  Arizona Cardinals / 3rd / 95th pick / 2009 28.31246     defense
## 5  Arizona Cardinals / 2nd / 63rd pick / 2009 31.32742     defense
## 6 Arizona Cardinals / 7th / 254th pick / 2009 37.36883     offense
##       Position_Type Position Drafted
## 1   backs_receivers       RB     Yes
## 2 defensive_lineman       DE     Yes
## 3 offensive_lineman       OG     Yes
## 4    defensive_back       FS     Yes
## 5       line_backer      OLB     Yes
## 6 offensive_lineman       OG     Yes
```

```
# manipulating data to become more usable
library(stringr)
library(dplyr)
```

```
##
## Attaching package: ’dplyr’
## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(readr)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

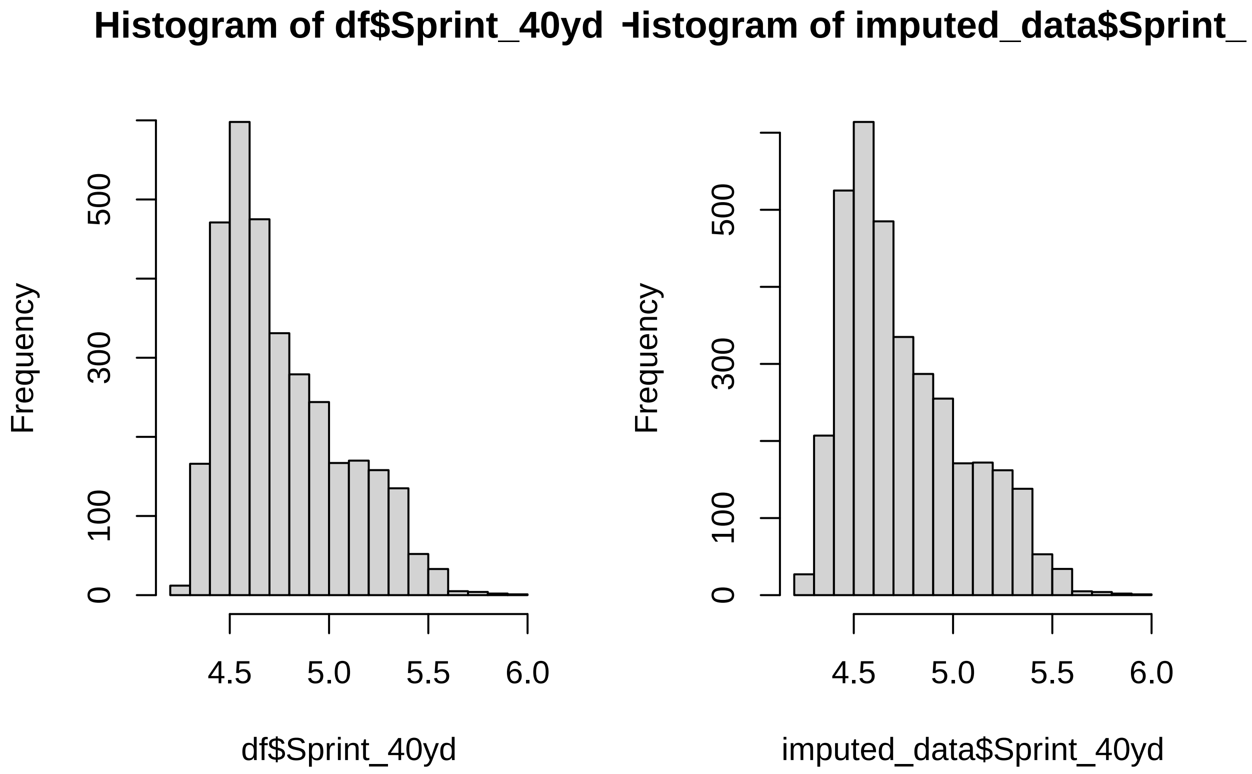
```
df = df %>% mutate(Team = str_split(Drafted..tm.rnd.yr., " / ", simplify = TRUE)[, 1]) # extract team d
df = df %>% mutate(Round = str_split(Drafted..tm.rnd.yr., " / ", simplify = TRUE)[, 2]) # extract round
df = df %>% mutate(Pick = str_split(Drafted..tm.rnd.yr., " / ", simplify = TRUE)[, 3]) # extract pick s
df$Pick = parse_number(df$Pick) # extract only the number of the pick, not the letters
df$Round = parse_number(df$Round)
df = df[,-13] # remove column with multiple columns worth of data
df$Pick = as.numeric(df$Pick)
df$Round = as.numeric(df$Round)
df$Player_Type = as.factor(df$Player_Type)
df$Position_Type = as.factor(df$Position_Type)
df$Position= as.factor(df$Position)
df$Drafted = as.factor(df$Drafted)
```

```
# need to preprocess the undrafted players. Set their value to -1
# Assuming your column names are "Drafted", "Round", and "Pick"
df$Round[df$Drafted == "No"] <- NA
df$Pick[df$Drafted == "No"] <- 400
```

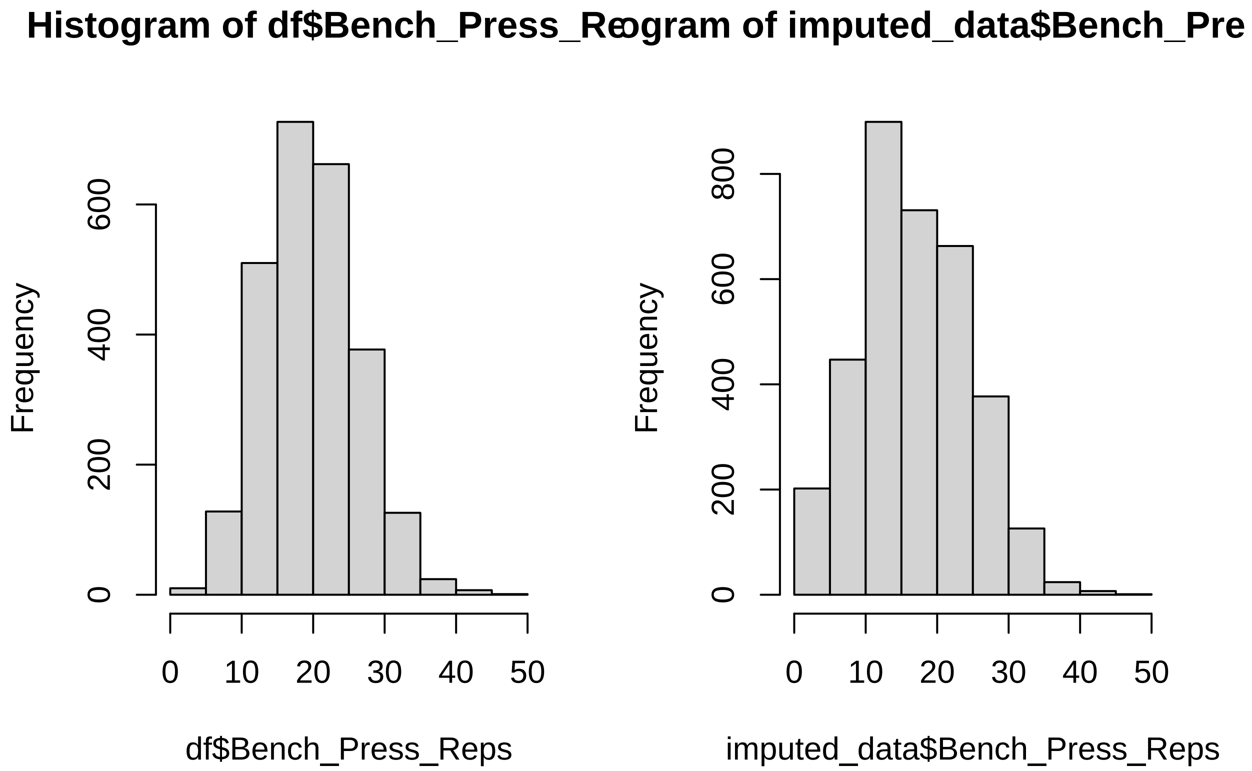
```
dim(df)
```

```
## [1] 3477   20
```

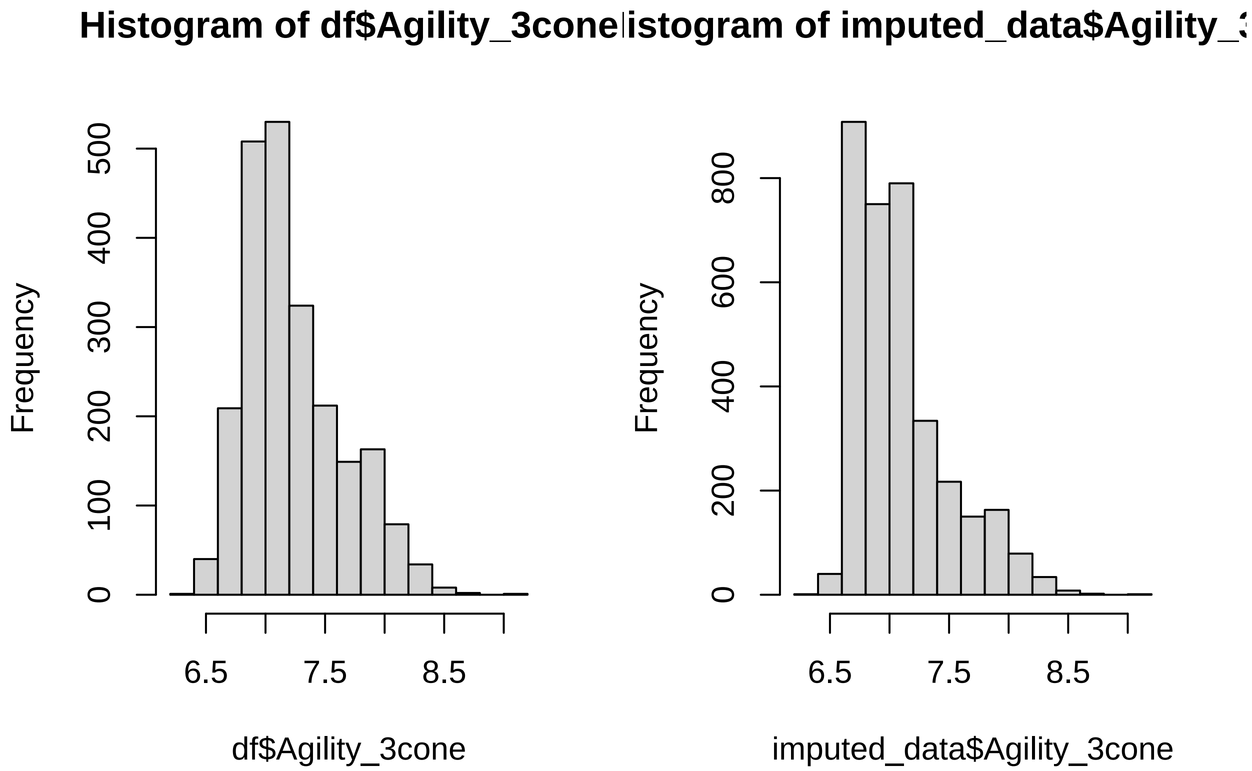
```
#compare distributions of original dataset and imputed dataset
par(mfrow=c(1,2))
hist(df$Sprint_40yd)
hist(imputed_data$Sprint_40yd)
```



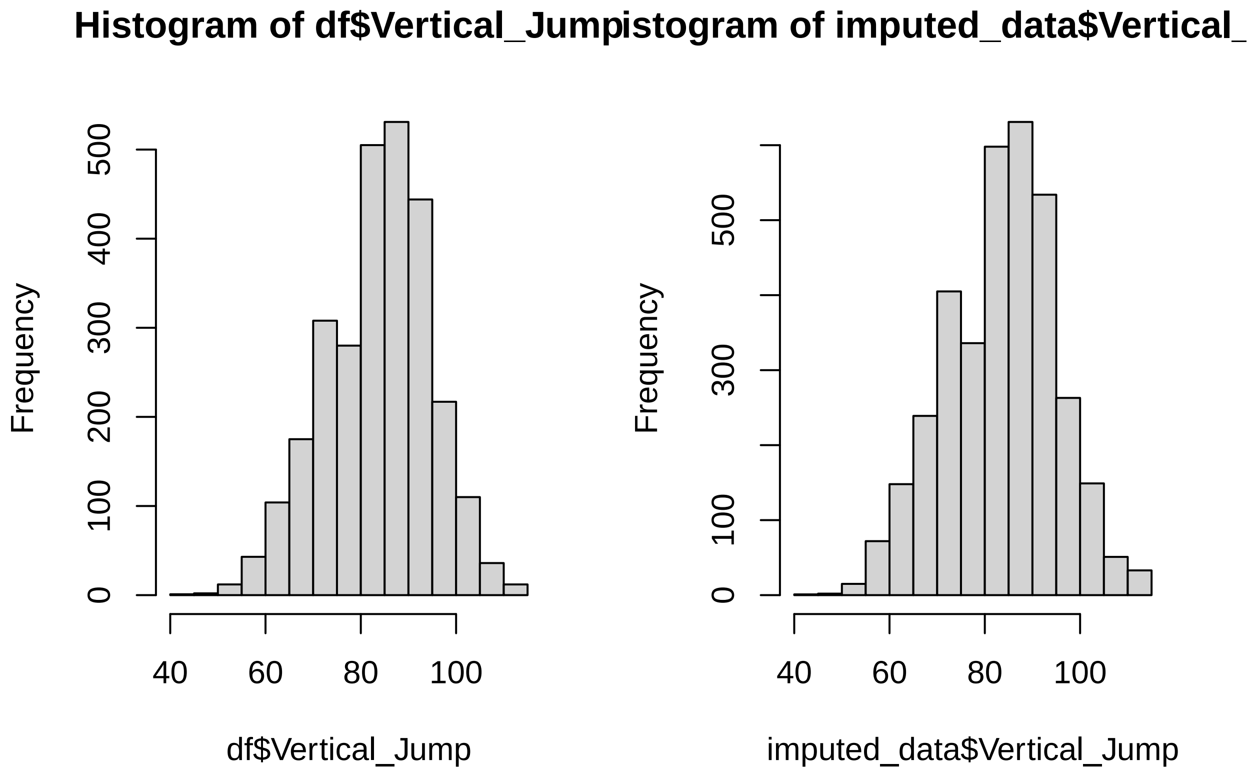
```
# compare distributions of original bench press and imputed
par(mfrow=c(1,2))
hist(df$Bench_Press_Reps)
hist(imputed_data$Bench_Press_Reps)
```



```r
#compare distributions of original 3 cone and imputed 3 cone
par(mfrow=c(1,2))
hist(df$Agility_3cone)
hist(imputed_data$Agility_3cone)
```



```
#compare distributions of original vertical jump and imputed vertical jump
par(mfrow=c(1,2))
hist(df$Vertical_Jump)
hist(imputed_data$Vertical_Jump)
```



Exploratory Data Analysis

```
summary(imputed_data$Drafted)
```

```
##   No  Yes
## 1223 2254
```

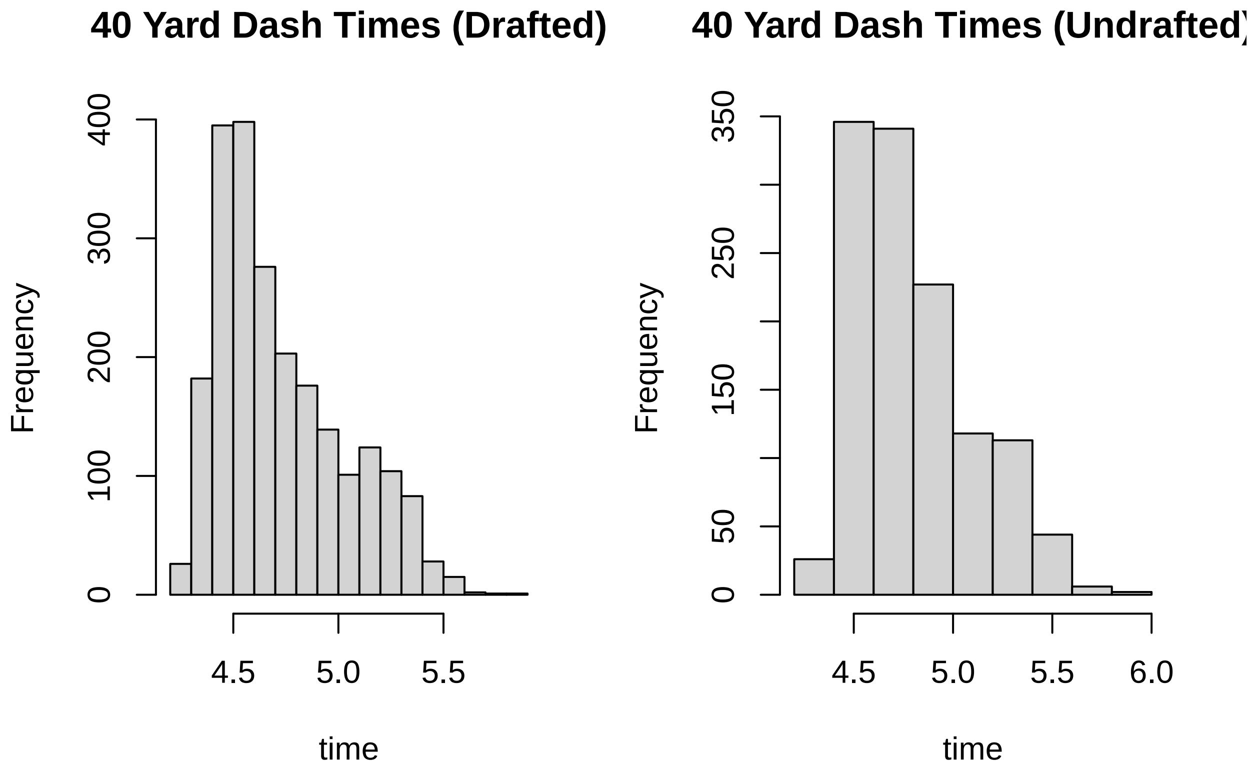
```
2254 + 12235
```

```
## [1] 14489
```

```
summary(imputed_data$Pick)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    85.0   177.0   215.5   400.0   400.0
```

```
# Compare distributions of 40 yard dash times between drafted and undrafted players
drafted = imputed_data[imputed_data$Drafted == "Yes", ]
undrafted = imputed_data[imputed_data$Drafted == "No", ]
par(mfrow=c(1,2))
hist(drafted$Sprint_40yd, main = "40 Yard Dash Times (Drafted)", xlab = "time")
hist(undrafted$Sprint_40yd, main = "40 Yard Dash Times (Undrafted)", xlab = "time")
```



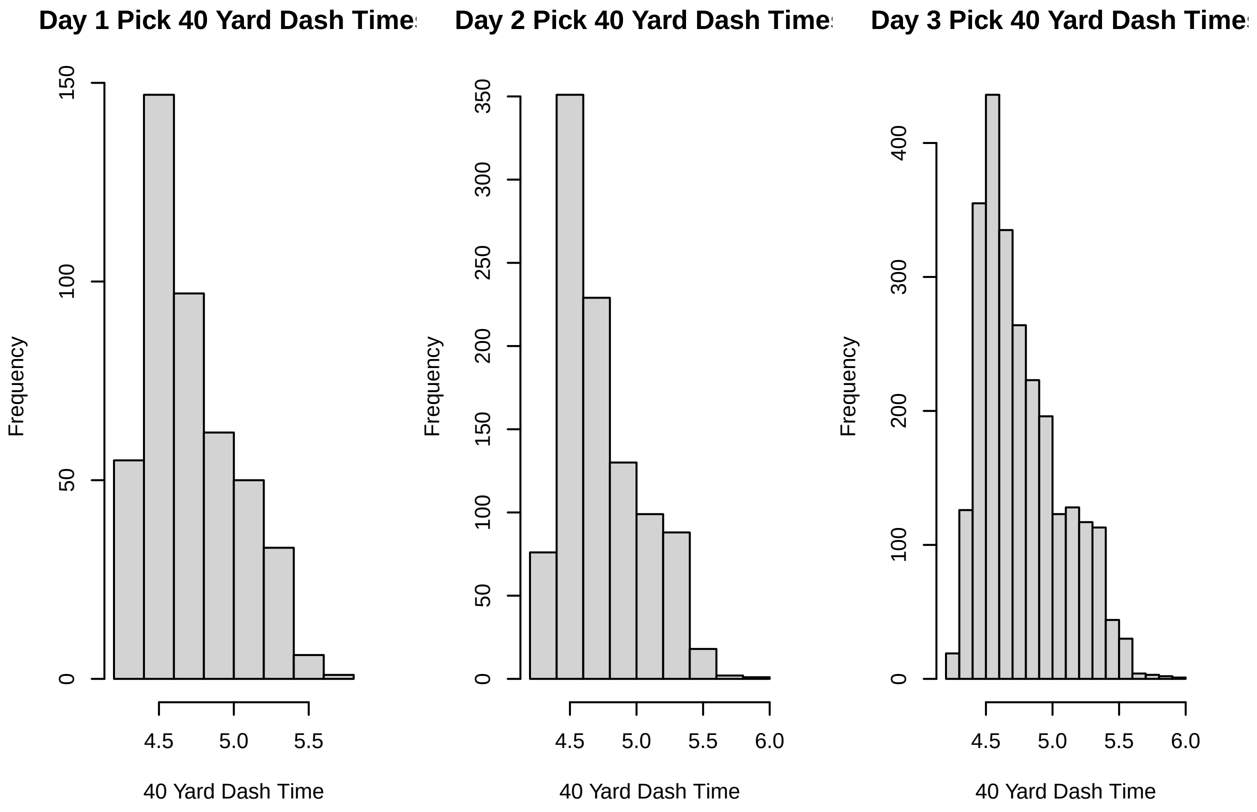
```
# compare means of 40 yard dash of drafted and undrafted players
mean(drafted$Sprint_40yd)
```

```
## [1] 4.732205
```

```
mean(undrafted$Sprint_40yd)
```

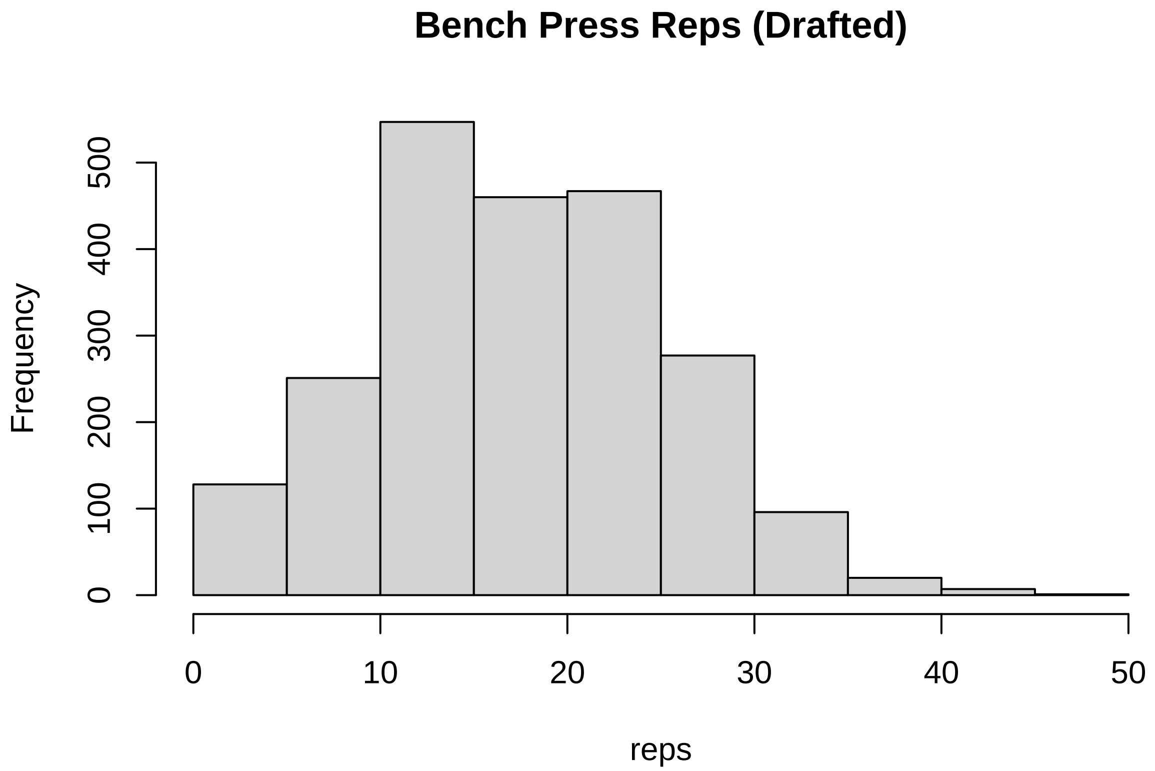
```
## [1] 4.807989
```

```
# comparing dash times of players by round range (day 1 day 2 day 3)
day1_pick = imputed_data[imputed_data$Round == 1,]
day2_pick = imputed_data[imputed_data$Round == 2 | imputed_data$Round == 3,]
day3_pick = imputed_data[imputed_data$Round == 3 | imputed_data$Round == 4 | imputed_data$Round == 5 |

par(mfrow=c(1,3))
hist(day1_pick$Sprint_40yd, main = "Day 1 Pick 40 Yard Dash Times", xlab = "40 Yard Dash Time")
hist(day2_pick$Sprint_40yd, main = "Day 2 Pick 40 Yard Dash Times", xlab = "40 Yard Dash Time")
hist(day3_pick$Sprint_40yd, main = "Day 3 Pick 40 Yard Dash Times", xlab = "40 Yard Dash Time")
```

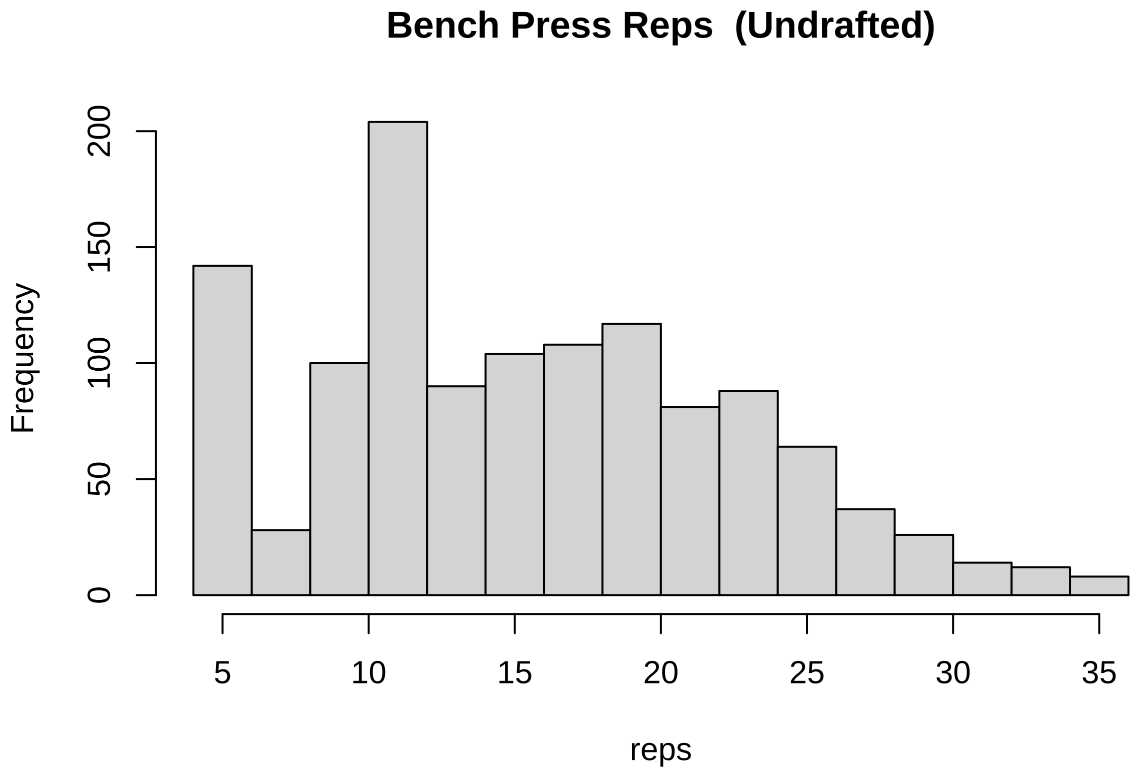


There seem to be more slower players picked later in the draft as opposed to earlier.

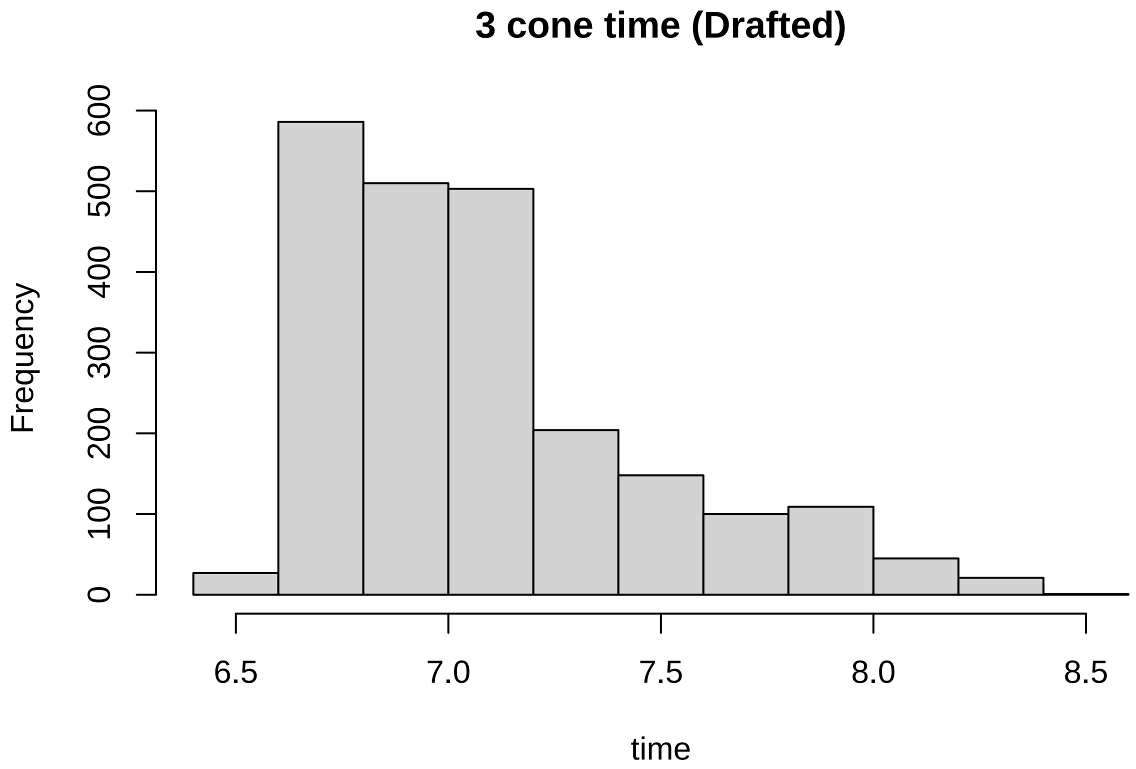
```
hist(drafted$Bench_Press_Reps, main = "Bench Press Reps (Drafted)", xlab = "reps")
```



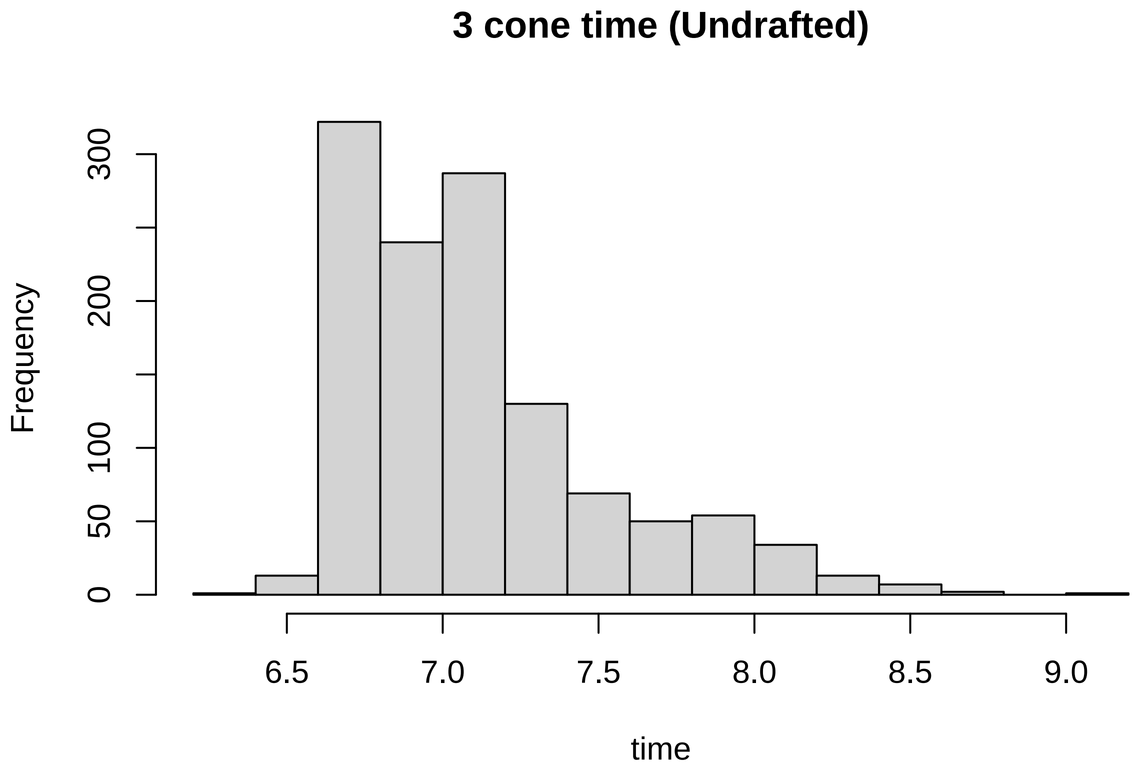
```
hist(undrafted$Bench_Press_Reps, main = "Bench Press Reps  (Undrafted)", xlab = "reps")
```



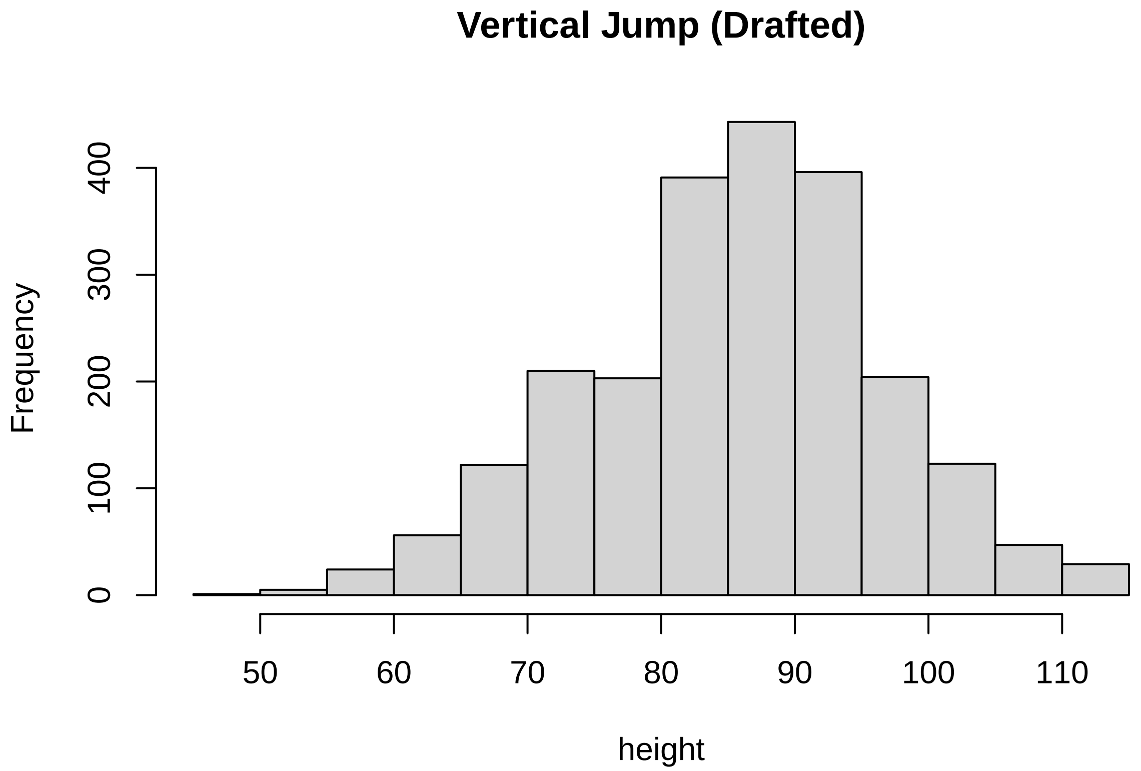
```
hist(drafted$Agility_3cone, main = "3 cone time (Drafted)", xlab = "time")
```



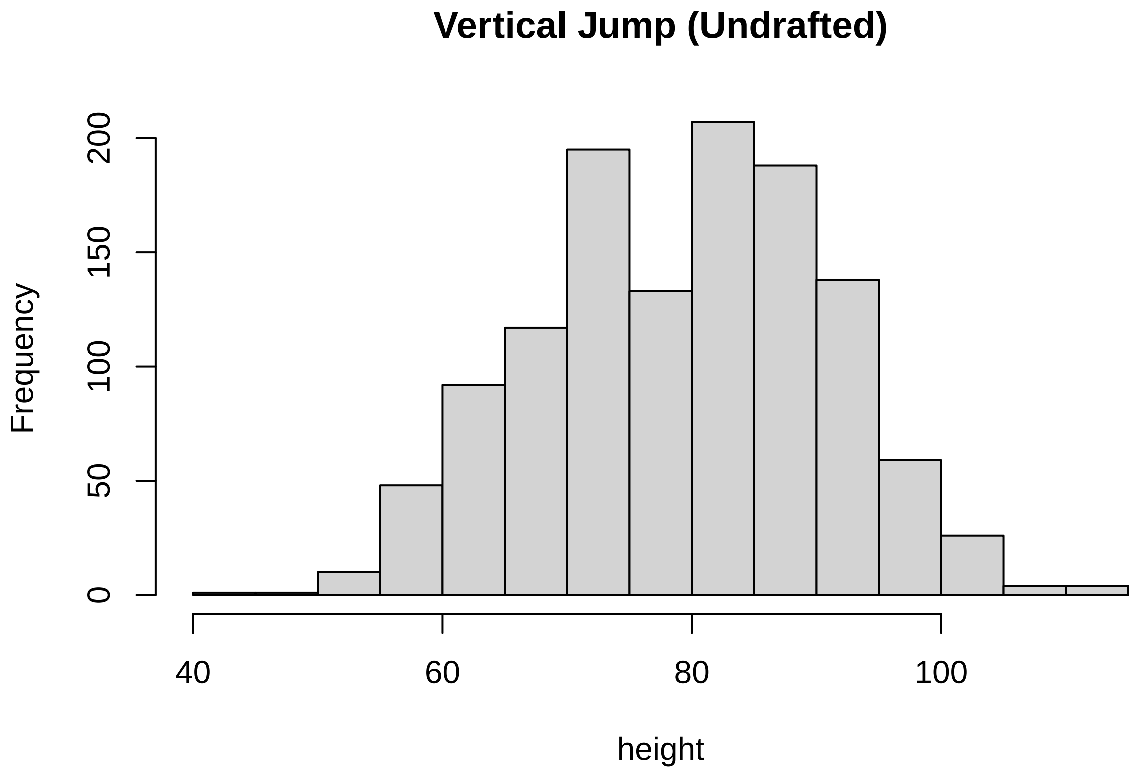
```
hist(undrafted$Agility_3cone, main = "3 cone time (Undrafted)", xlab = "time")
```



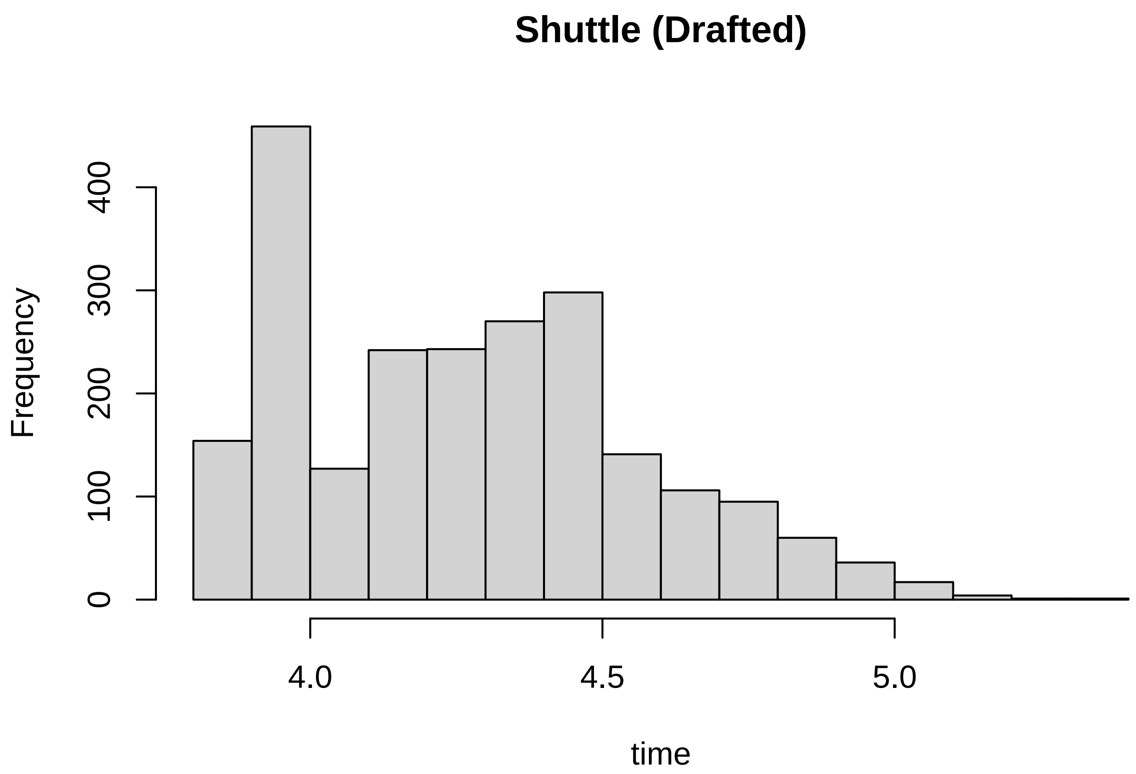
```r
hist(drafted$Vertical_Jump, main = "Vertical Jump (Drafted)", xlab = "height")
```



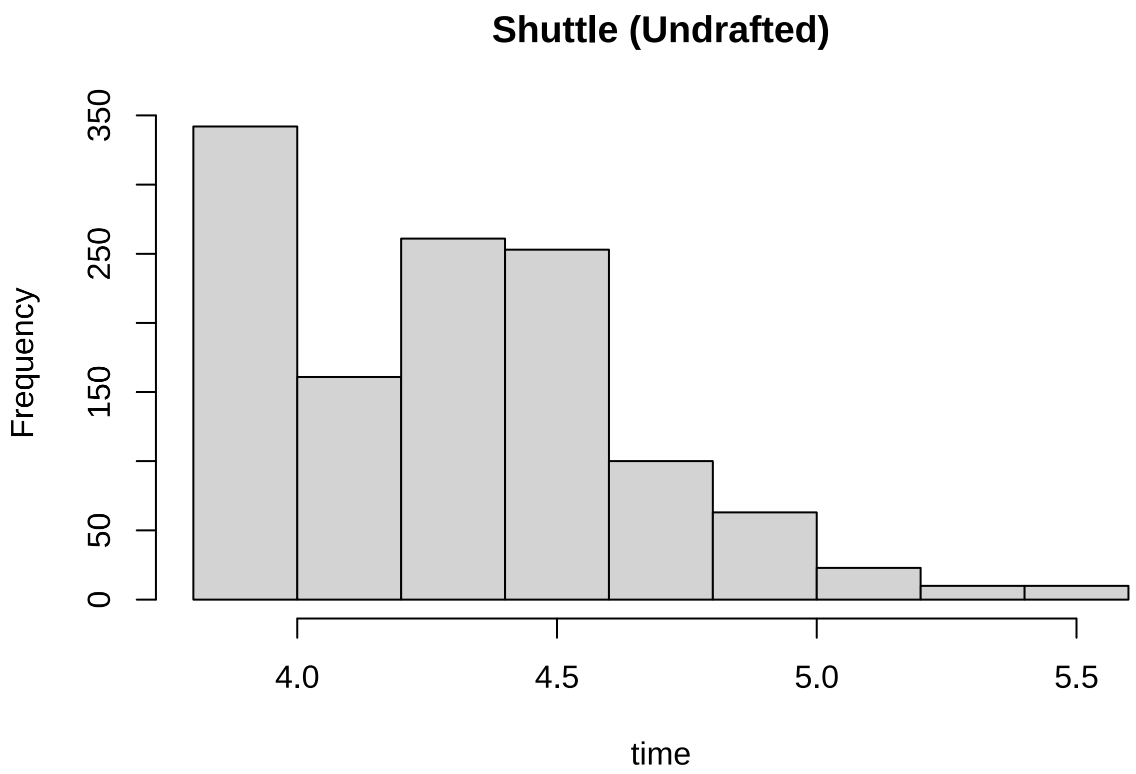
```
hist(undrafted$Vertical_Jump, main = "Vertical Jump (Undrafted)", xlab = "height")
```



```r
hist(drafted$Shuttle, main = "Shuttle (Drafted)", xlab = "time")
```



```
hist(undrafted$Shuttle, main = "Shuttle (Undrafted)", xlab = "time")
```



```
# compare means of dash times by day
mean(day1_pick$Sprint_40yd)
```

```
## [1] 4.717206
```

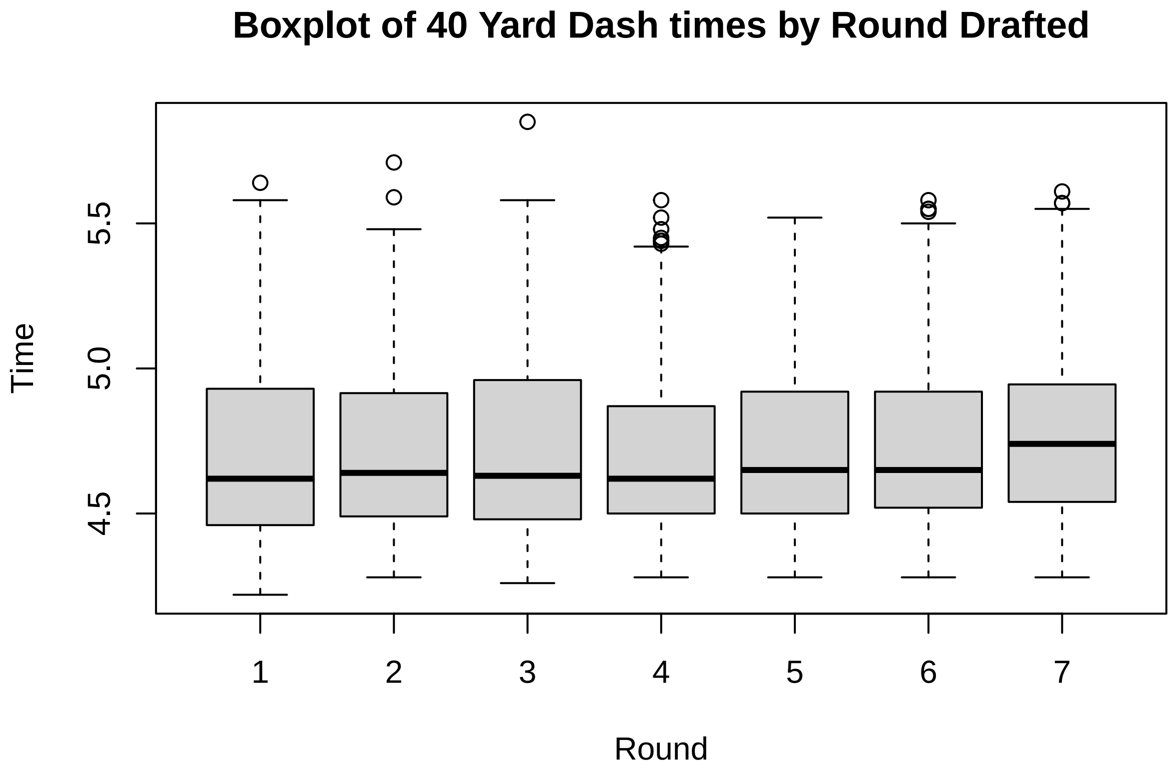
```
mean(day2_pick$Sprint_40yd)
```

```
## [1] 4.739507
```

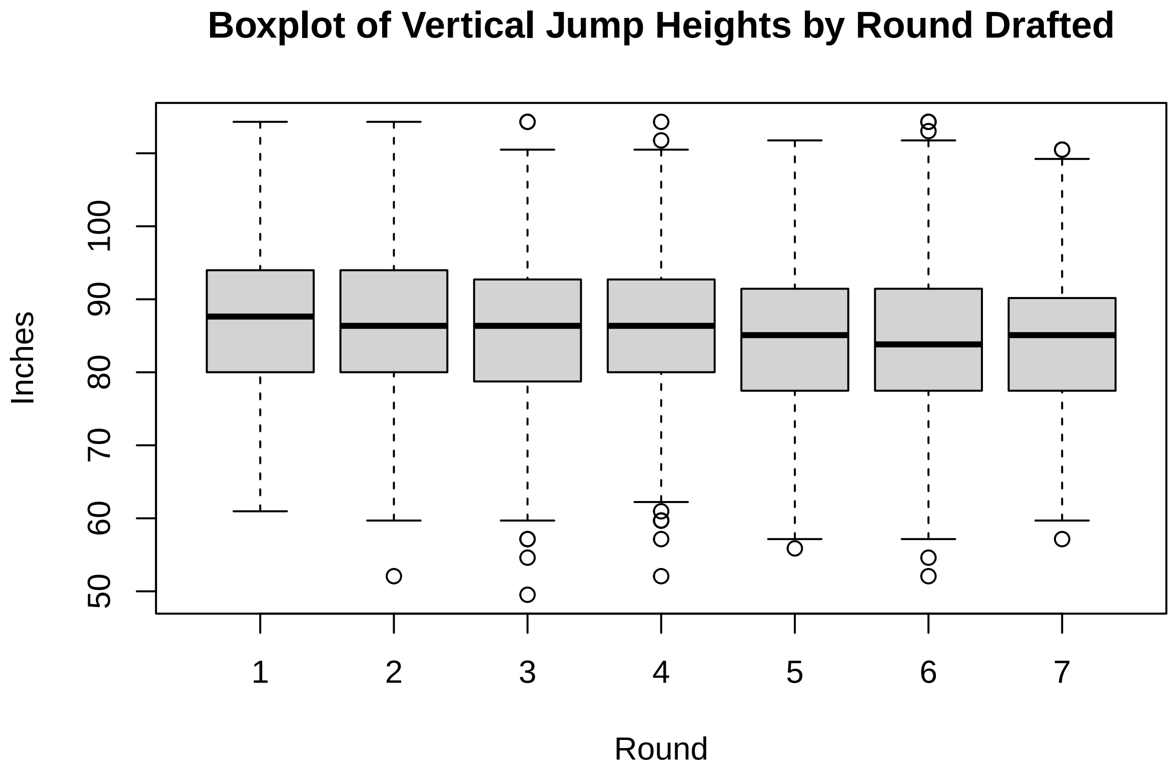
```
mean(day3_pick$Sprint_40yd)
```

```
## [1] 4.77636
```

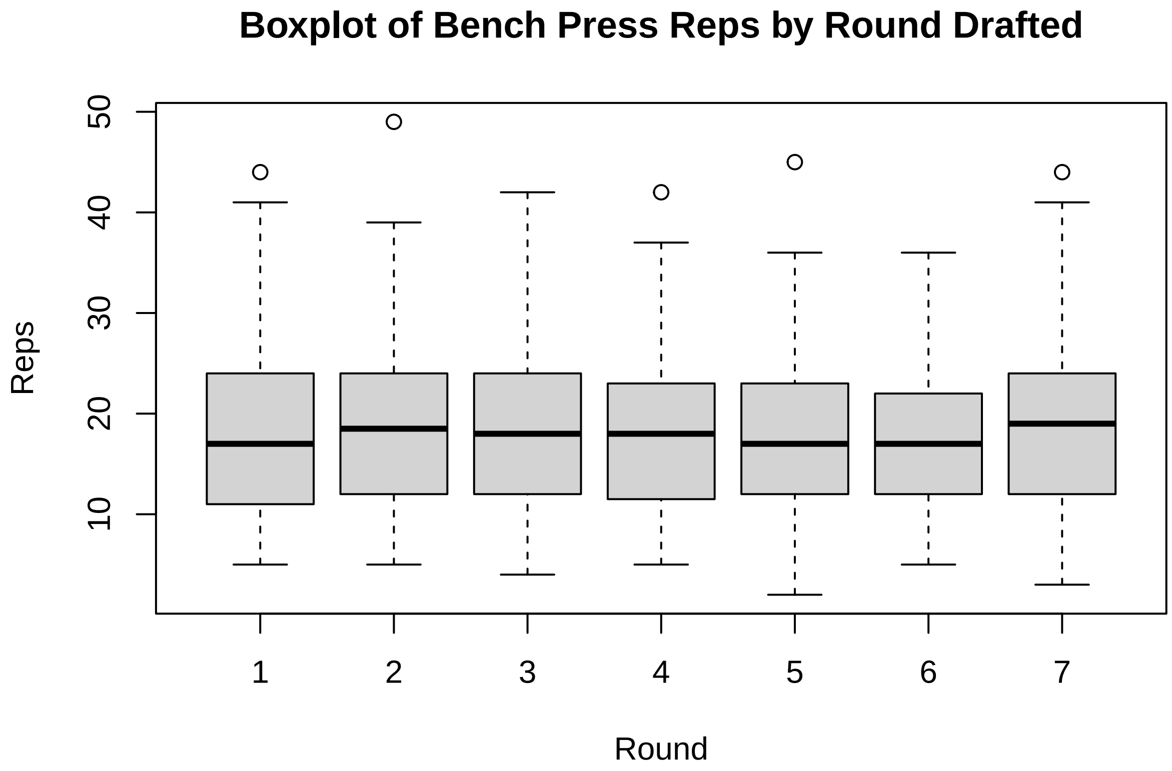
```
library(lattice)
boxplot(Sprint_40yd ~ Round, data=drafted, main="Boxplot of 40 Yard Dash times by Round Drafted", ylab =
```



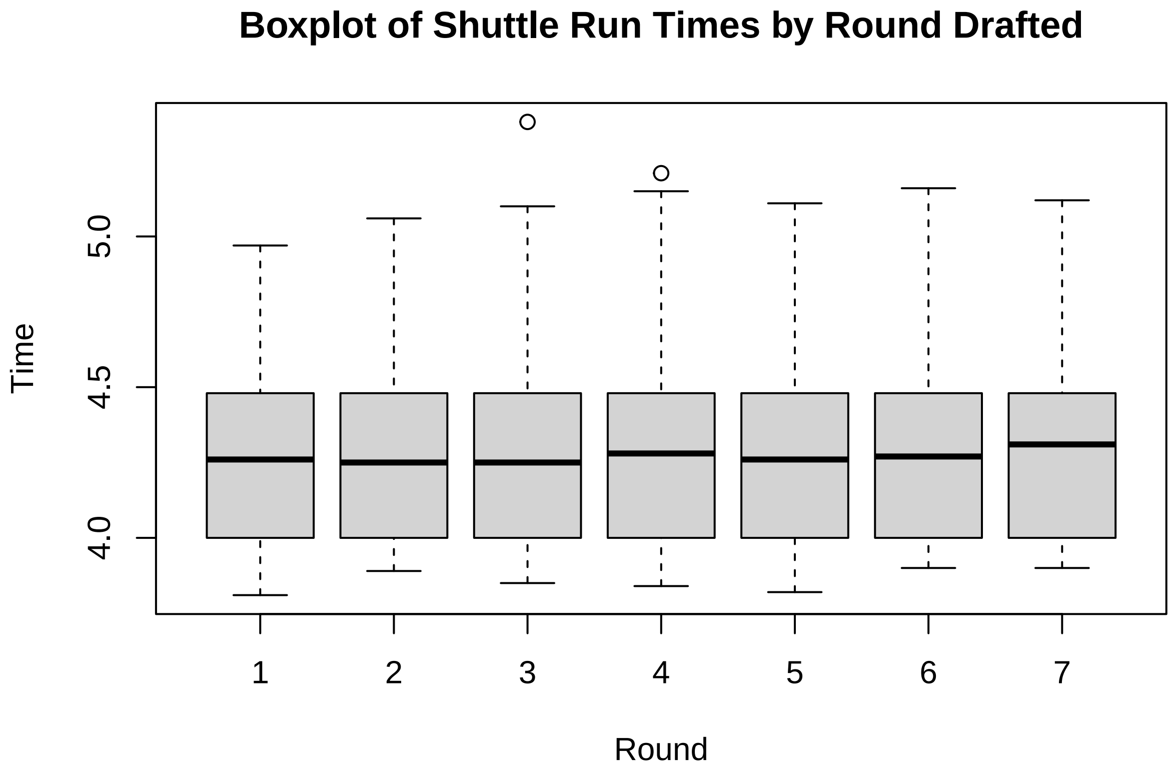
```
boxplot(Vertical_Jump ~ Round, data=drafted, main="Boxplot of Vertical Jump Heights by Round Drafted", 
```



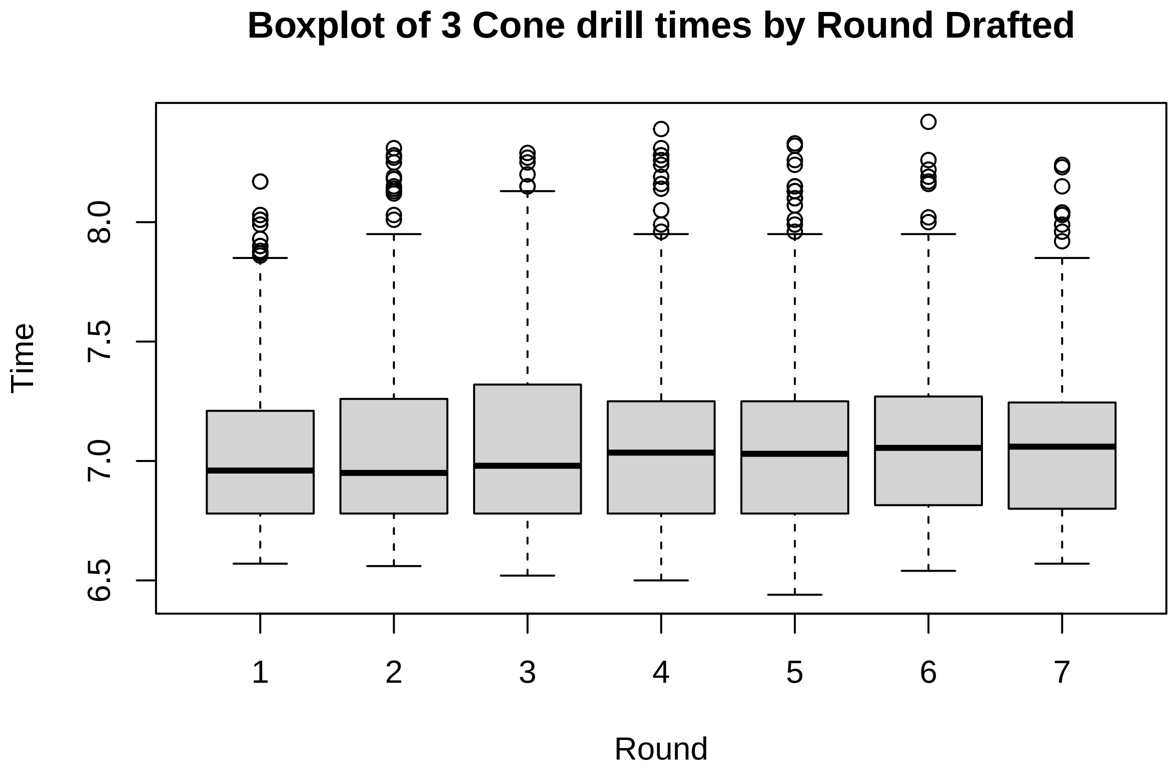
```
boxplot(Bench_Press_Reps ~ Round, data = drafted, main="Boxplot of Bench Press Reps by Round Drafted", 
```



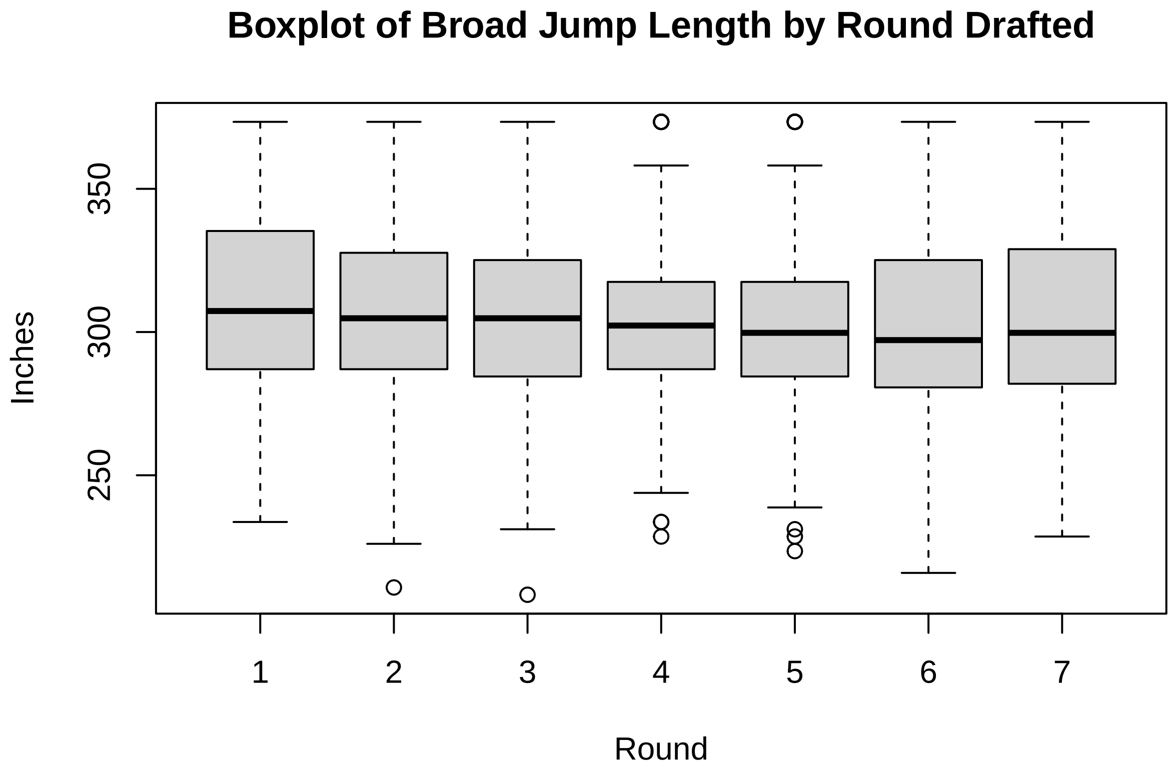
```
boxplot(Shuttle ~ Round, data=drafted, main="Boxplot of Shuttle Run Times by Round Drafted", ylab = "Ti
```



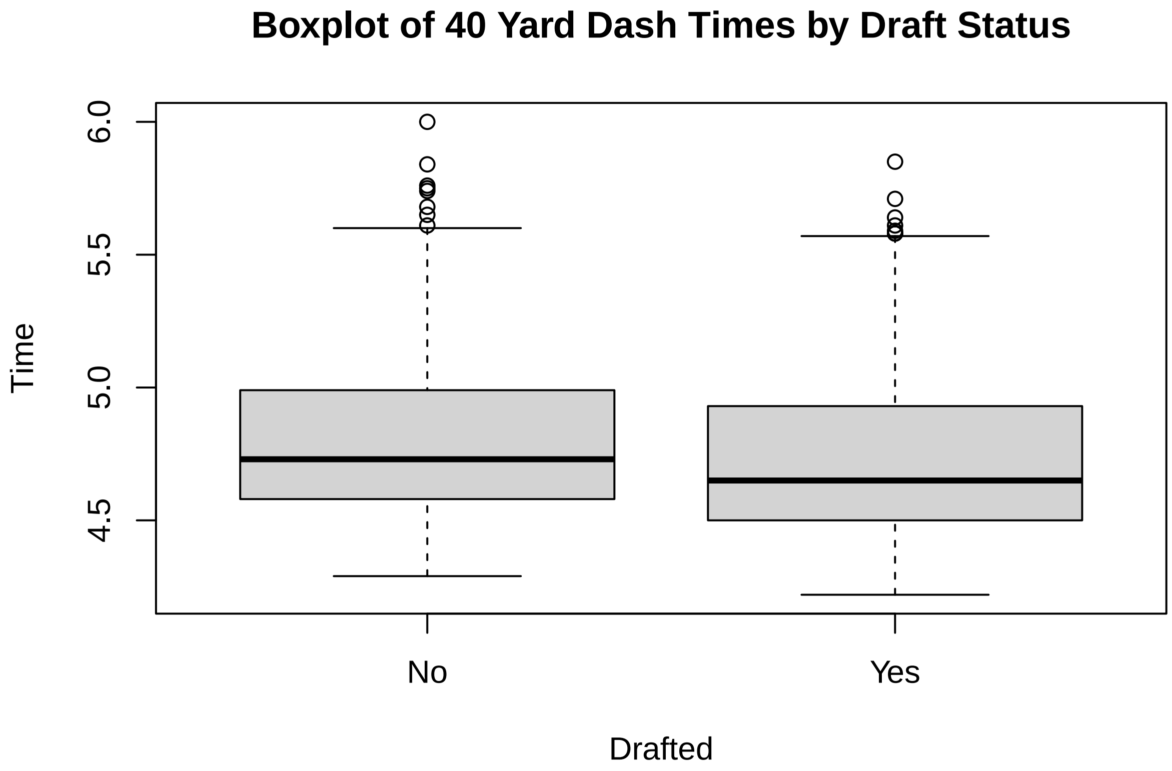
```
boxplot(Agility_3cone ~ Round, data = drafted, main = "Boxplot of 3 Cone drill times by Round Drafted",
```



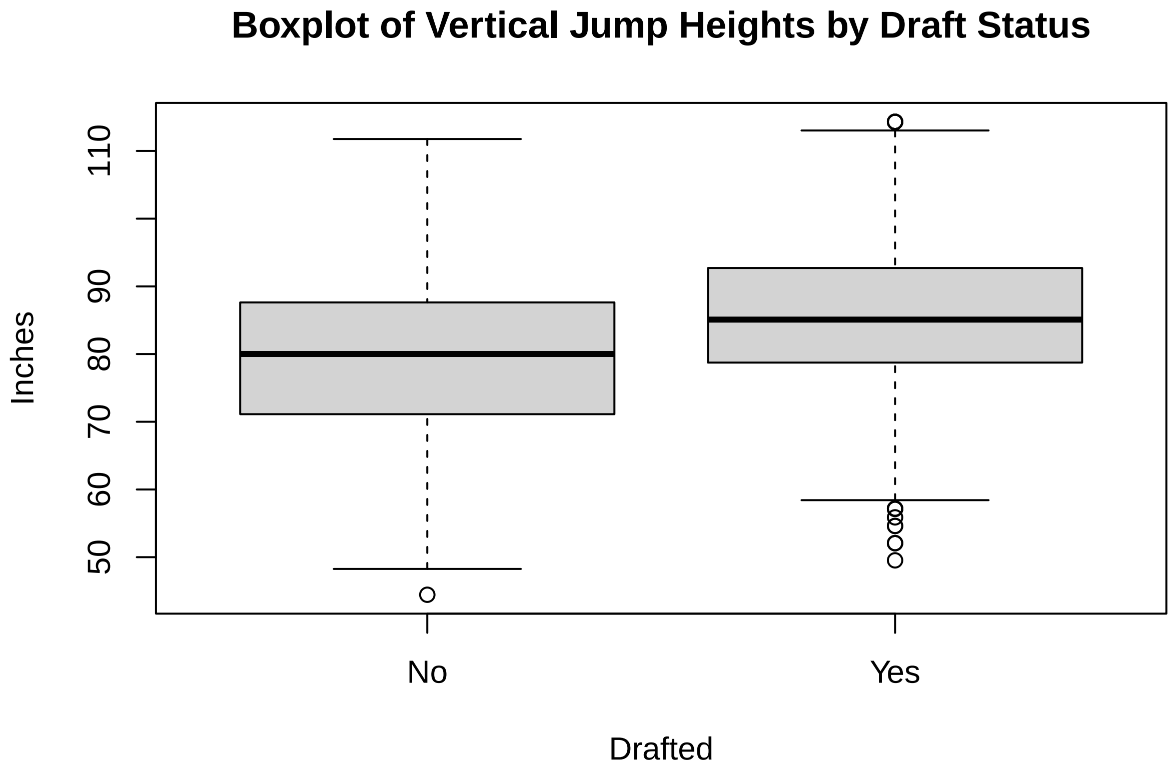
```
boxplot(Broad_Jump ~ Round, data = drafted, main ="Boxplot of Broad Jump Length by Round Drafted", ylab
```



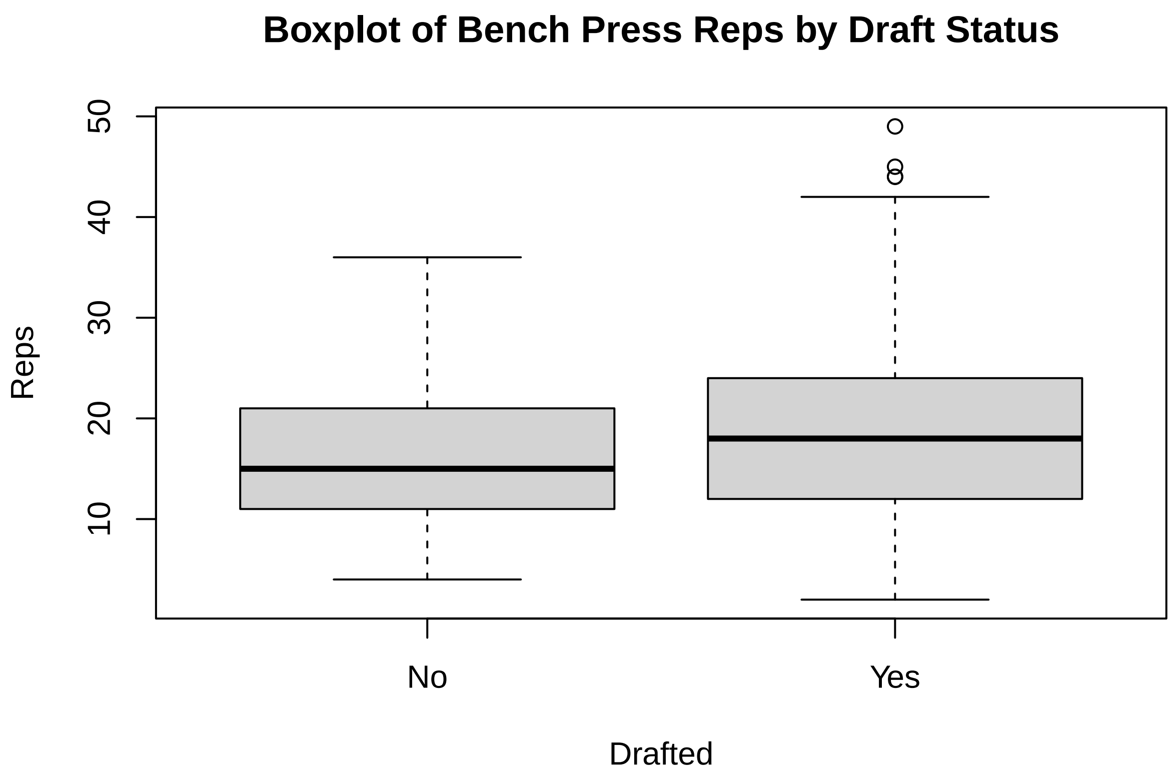
```
boxplot(Sprint_40yd ~ Drafted, data=imputed_data, main = "Boxplot of 40 Yard Dash Times by Draft Status
```



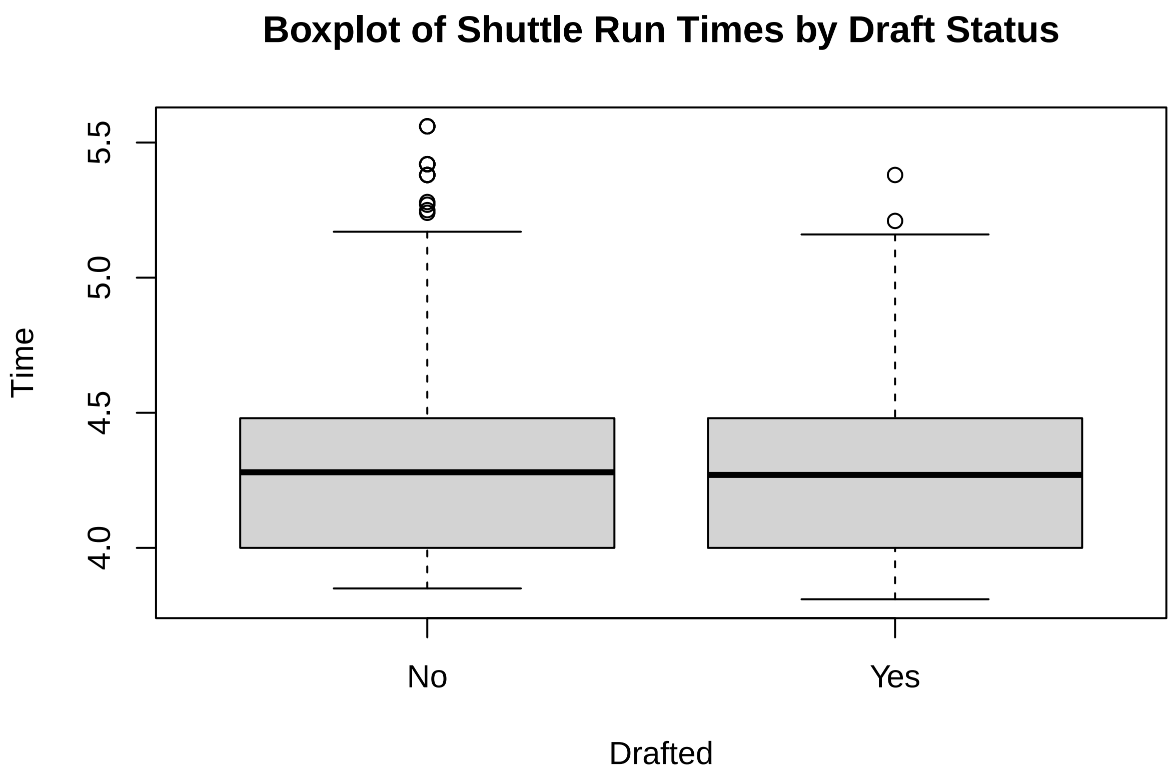
```
boxplot(Vertical_Jump ~ Drafted, data=imputed_data, main = "Boxplot of Vertical Jump Heights by Draft S
```



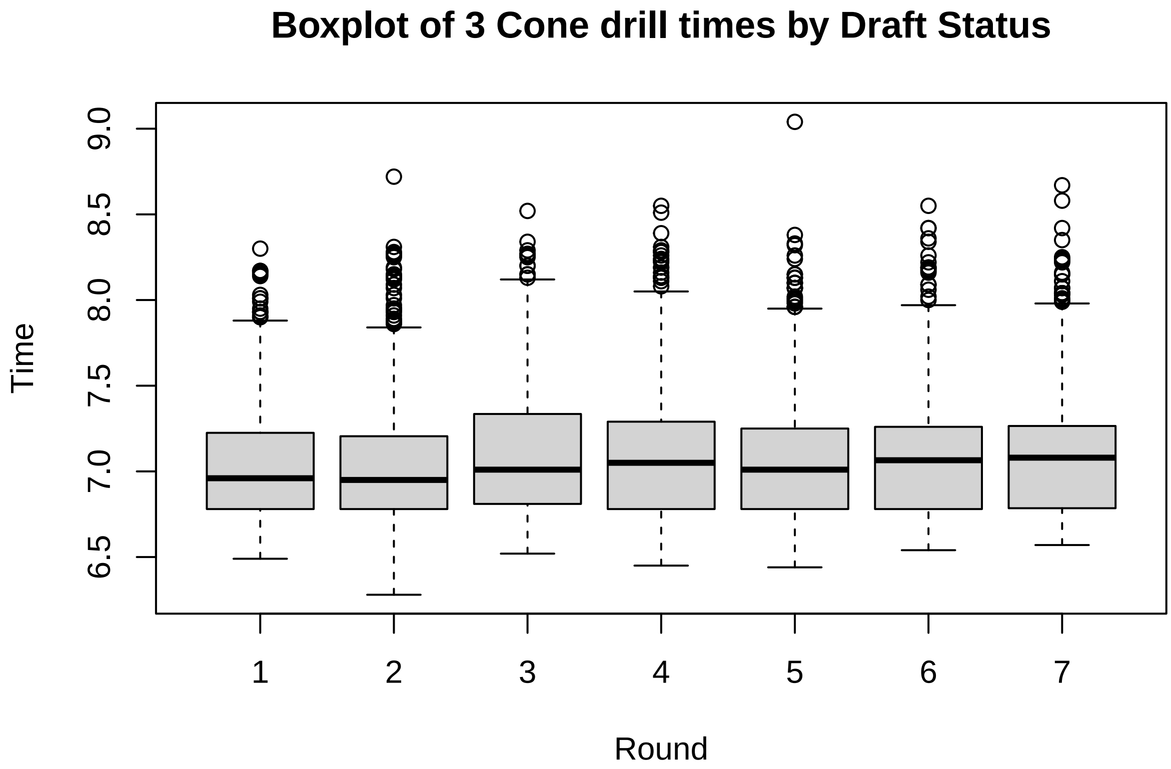
```
boxplot(Bench_Press_Reps ~ Drafted, data=imputed_data, main = "Boxplot of Bench Press Reps by Draft Sta
```



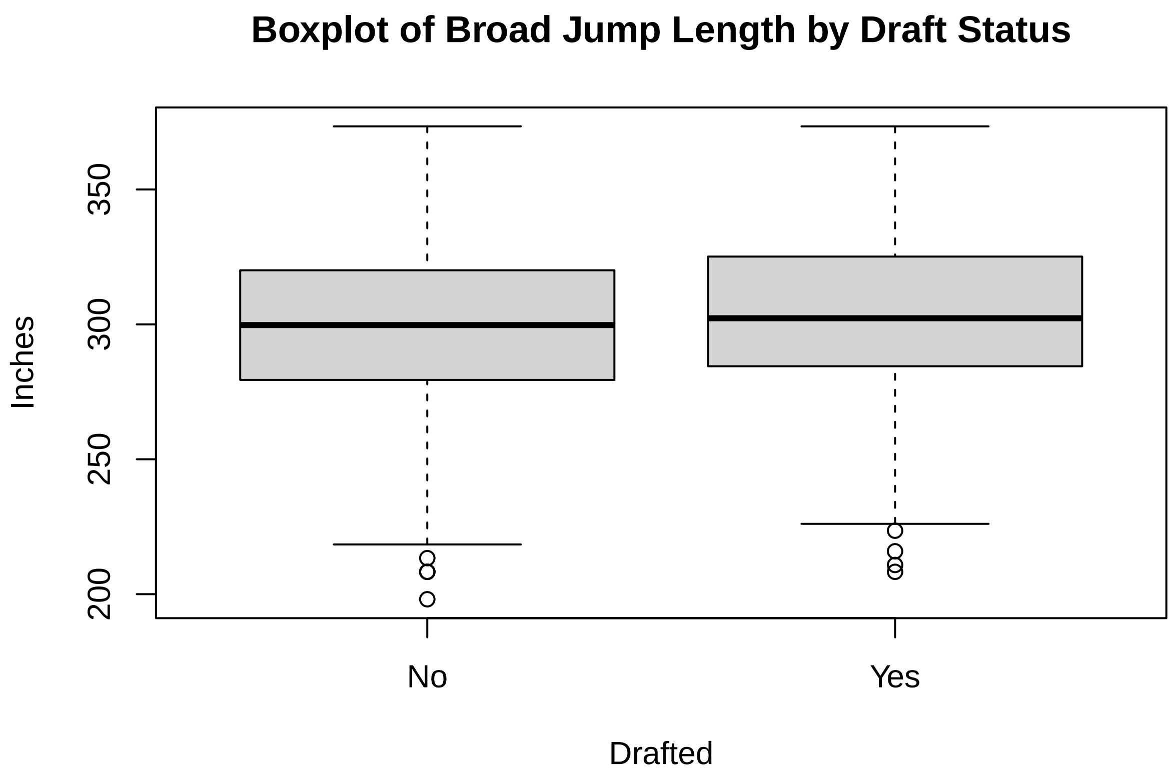
```
boxplot(Shuttle ~ Drafted, data=imputed_data, main = "Boxplot of Shuttle Run Times by Draft Status", yla
```



```
boxplot(Agility_3cone ~ Round, data = imputed_data, main = "Boxplot of 3 Cone drill times by Draft Statu
```

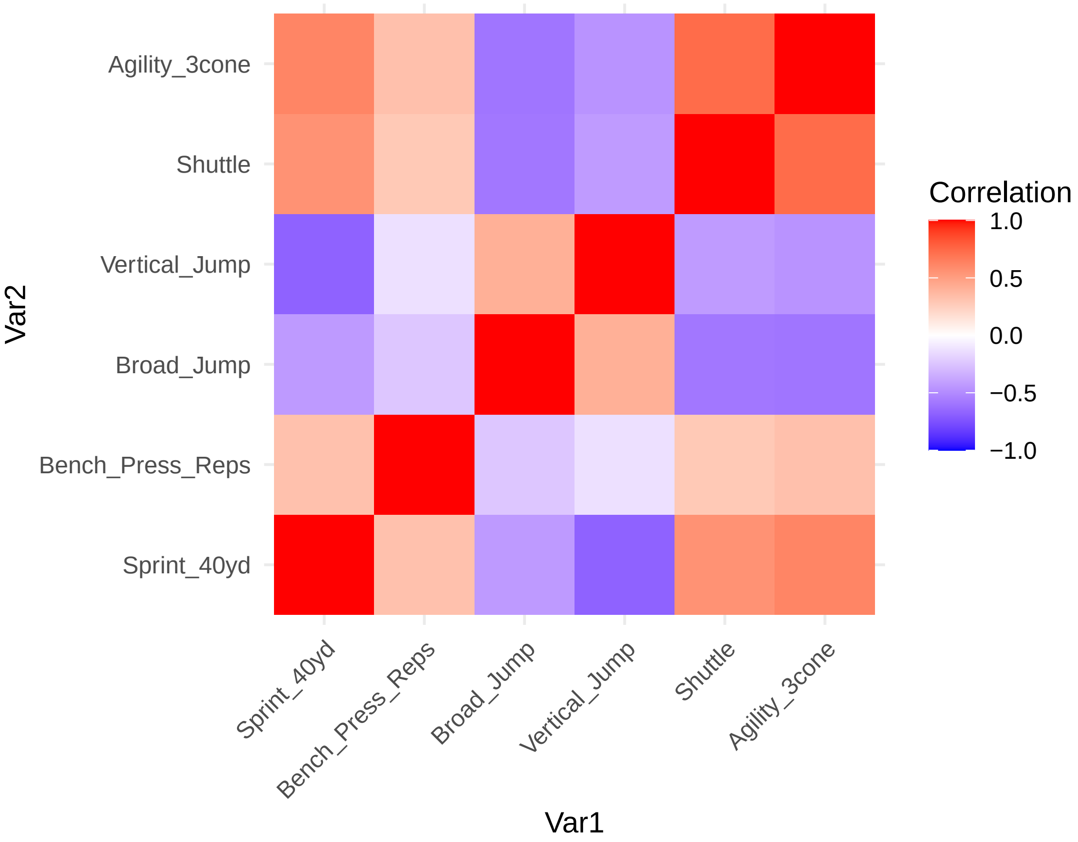


```
boxplot(Broad_Jump ~ Drafted, data = imputed_data, main = "Boxplot of Broad Jump Length by Draft Status
```



```
library(ggplot2)
library(reshape2)
sample = sample(nrow(imputed_data), 0.8*nrow(imputed_data))
train = imputed_data[sample,]
test = imputed_data[-sample,]

numeric_features <- train[, c("Sprint_40yd", "Bench_Press_Reps", "Broad_Jump", "Vertical_Jump", "Shuttl
cor_matrix <- cor(numeric_features, use = "pairwise.complete.obs")
```

```
heatmap_data <- melt(cor_matrix)
colnames(heatmap_data) <- c("Var1", "Var2", "value")
ggplot(data = heatmap_data, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1), space
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  coord_fixed()
```



Some of the features are correlated, so we should consider this when doing our analysis.

Analysis

Logistic Regression and Linear Regression

For this analysis, there are three parts. The first part is a logistic regression to predict whether a player will be drafted or not based on their combine performance. From there, we will use a multiple linear regression to predict how high a player will be picked using only the players that were actually drafted, since an undrafted player cannot be assigned a pick value. The purpose of these two models is to first predict whether a player will be drafted at all, and if they are predicted to be drafted, to then predict their draft pick value given their combine numbers.

The third method will be a Random Forest.

Logistic Regression

```
# use all the drills
logit_model <- glm(Drafted ~ Sprint_40yd + Bench_Press_Reps + Broad_Jump + Vertical_Jump + Shuttle + Agi
summary(logit_model)
```

```
##
## Call:
```

```
## glm(formula = Drafted ~ Sprint_40yd + Bench_Press_Reps + Broad_Jump + 
##     Vertical_Jump + Shuttle + Agility_3cone, family = "binomial", 
##     data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.2243  -1.1785   0.6984   0.9209   1.7533  
## 
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)    
## (Intercept)      -8.272757   1.666361  -4.965 6.89e-07 ***
## Sprint_40yd       0.090761   0.216161   0.420  0.67457    
## Bench_Press_Reps  0.034862   0.006071   5.743 9.31e-09 ***
## Broad_Jump       -0.001074   0.001549  -0.694  0.48786    
## Vertical_Jump     0.065737   0.005520  11.908  < 2e-16 ***
## Shuttle          -0.280695   0.213773  -1.313  0.18917    
## Agility_3cone     0.559792   0.174907   3.201  0.00137 ** 
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 3610.5  on 2780  degrees of freedom
## Residual deviance: 3339.5  on 2774  degrees of freedom
## AIC: 3353.5
## 
## Number of Fisher Scoring iterations: 4
```

```
predicted_probs = predict(logit_model, newdata=test, type="response")
test_pred_class = ifelse(predicted_probs > 0.5, "Yes", "No")
confusionMatrix(table(test_pred_class, test$Drafted))
```

```
## Confusion Matrix and Statistics
## 
##                
## test_pred_class  No Yes
##             No   76  43
##             Yes 166 411
##                                           
##                Accuracy : 0.6997          
##                  95% CI : (0.6641, 0.7336)
##     No Information Rate : 0.6523          
##     P-Value [Acc > NIR] : 0.004497        
##                                           
##                   Kappa : 0.2489          
##                                           
##  Mcnemar’s Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 0.3140          
##             Specificity : 0.9053          
##          Pos Pred Value : 0.6387          
##          Neg Pred Value : 0.7123          
##              Prevalence : 0.3477          
##          Detection Rate : 0.1092          
##    Detection Prevalence : 0.1710
##       Balanced Accuracy : 0.6097
##
##        'Positive' Class : No
##
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

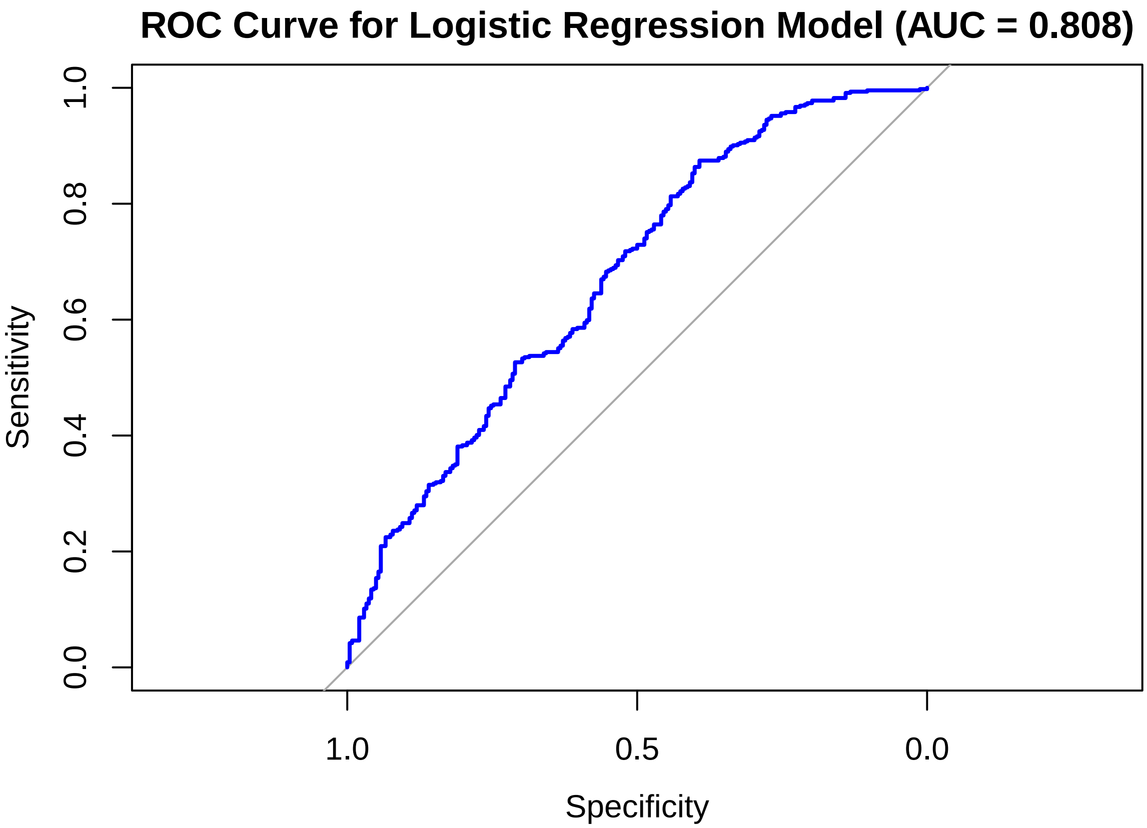
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Step 3: Predict probabilities on the testing set
#predicted_probs <- predict(logit_model, newdata = test, type = "response")

# Step 4: Create an ROC curve using true values and predicted probabilities
roc_obj <- roc(test$Drafted, predicted_probs)
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
# Plot the ROC curve
plot(roc_obj, main = "ROC Curve for Logistic Regression Model (AUC = 0.808)", col = "blue", lwd = 2)
```



```
auc(roc_obj)
```

```
## Area under the curve: 0.6733
```

Linear Regression

```
predicted_draft_status <- predict(logit_model, newdata = test, type = "response")
predicted_draft_status <- ifelse(predicted_draft_status >= 0.5, "Yes", "No") # Threshold at 0.5, adjust
drafted_data <- test[predicted_draft_status == "Yes", ]

# Step 5: Fit a linear regression model on the filtered data (assuming 'Pick' is the response variable
linear_model <- lm(Pick ~ Sprint_40yd + Bench_Press_Reps + Broad_Jump + Vertical_Jump + Shuttle + Agili
summary(linear_model)
```

```
##
## Call:
## lm(formula = Pick ~ Sprint_40yd + Bench_Press_Reps + Broad_Jump +
##     Vertical_Jump + Shuttle + Agility_3cone, data = drafted_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -259.62 -108.41  -36.27  150.86  264.33
##
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      795.8804   225.4062   3.531 0.000448 ***
## Sprint_40yd        1.9419    28.8727   0.067 0.946400    
## Bench_Press_Reps  -3.6141     0.8441  -4.282 2.18e-05 ***
## Broad_Jump        -0.7432     0.2405  -3.091 0.002095 ** 
## Vertical_Jump     -1.9664     0.8040  -2.446 0.014751 *  
## Shuttle          -43.8832    32.5405  -1.349 0.178011    
## Agility_3cone      6.6403    26.6771   0.249 0.803518    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 136 on 570 degrees of freedom
## Multiple R-squared:  0.06487,    Adjusted R-squared:  0.05503 
## F-statistic: 6.591 on 6 and 570 DF,  p-value: 9.528e-07
```

```
test_pred_linear <- predict(linear_model, newdata = test)
linear_regression_mse <- mean((test$Pick - test_pred_linear)^2)
linear_regression_rmse <- sqrt(linear_regression_mse)
cat("RMSE for Linear Regression (Drafted Players): ", linear_regression_rmse, "\n")
```

```
## RMSE for Linear Regression (Drafted Players):  138.827
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
## 
## Attaching package: 'randomForest'
```

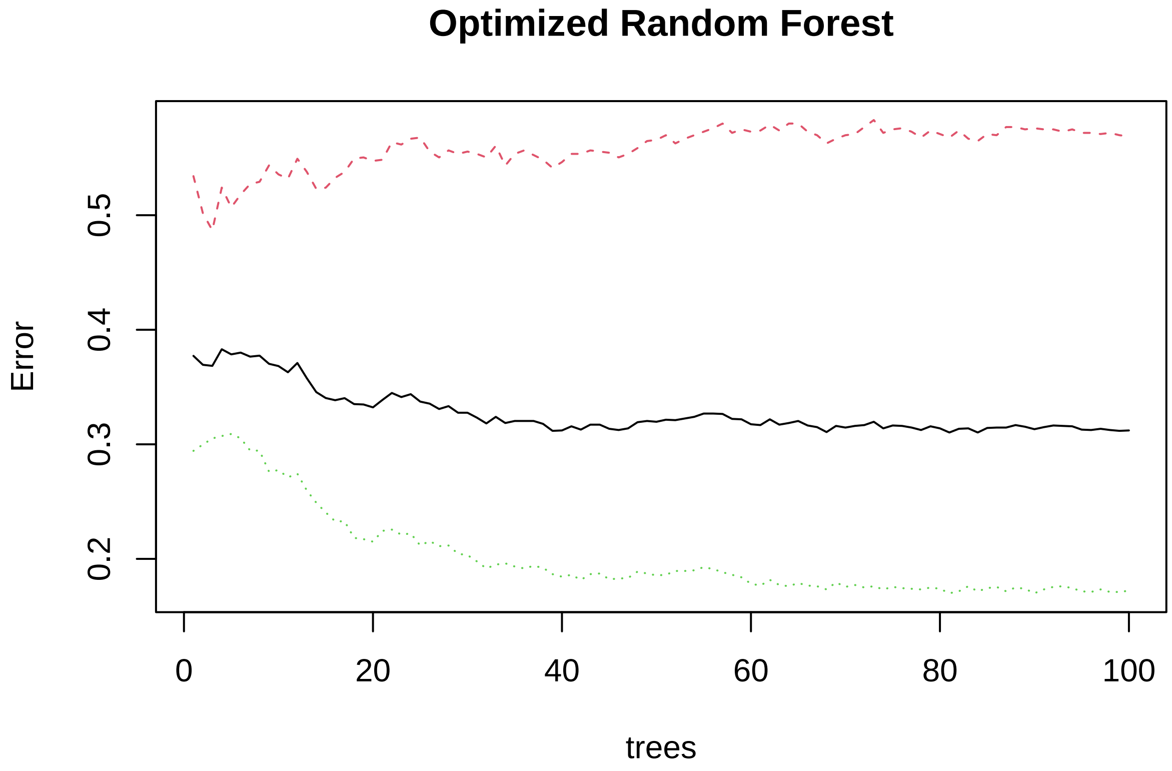
```
## The following object is masked from 'package:ggplot2':
## 
##     margin
```

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

```
train_control <- trainControl(method = "cv", number = 5, search = "grid")
rf_grid <- expand.grid(.mtry = seq(1, 6, by = 1))
optimized_rf1 <- randomForest(Drafted ~ Sprint_40yd + Bench_Press_Reps + Broad_Jump + Vertical_Jump + S
print(optimized_rf1)
```

```
##
## Call:
##  randomForest(formula = Drafted ~ Sprint_40yd + Bench_Press_Reps +      Broad_Jump + Vertical_Jump +
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 31.21%
## Confusion matrix:
##      No  Yes class.error
## No  423  558   0.5688073
## Yes 310 1490   0.1722222
```

```
plot(optimized_rf1, main = "Optimized Random Forest")
```

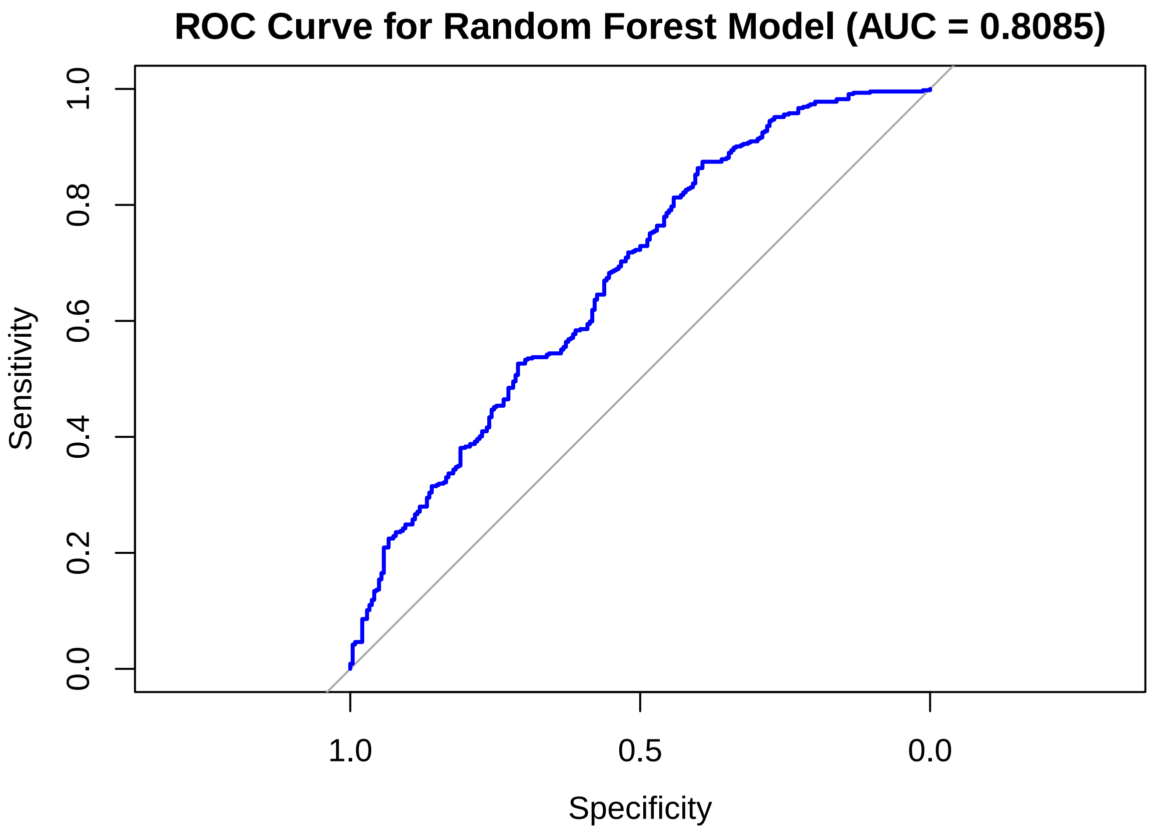


```
library(pROC)
# Predict probabilities on the testing set
predicted_probs_rf1 <- predict(optimized_rf1, newdata = test, type = "prob")[, 2] # [, 2] to select the

# Create an ROC curve using true values and predicted probabilities
roc_obj <- roc(test$Drafted, predicted_probs)
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plot(roc_obj, main = "ROC Curve for Random Forest Model (AUC = 0.8085)", col = "blue", lwd = 2)
```



```
auc_value <- auc(roc_obj)
print(auc_value)
```

```
## Area under the curve: 0.6733
```

```
pred_rf1 = ifelse(predicted_probs_rf1 >= 0.5, "Yes", "No")
confusionMatrix(table(pred_rf1, test$Drafted))
```

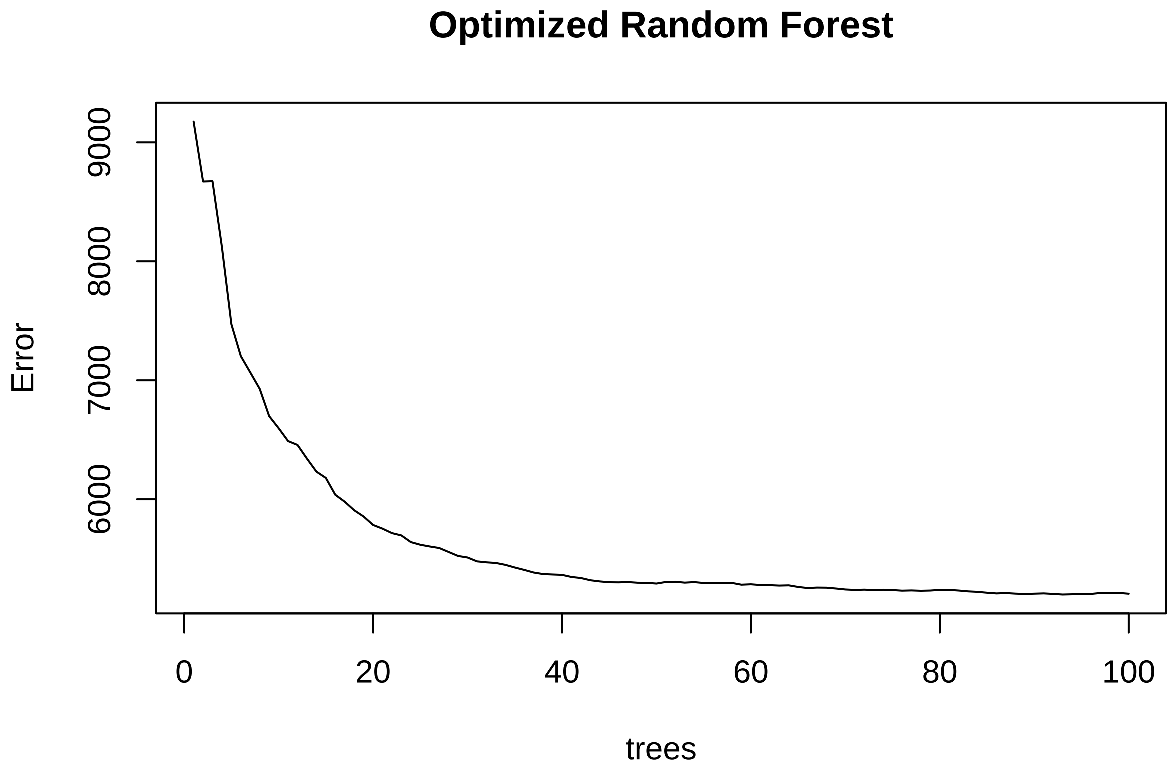
```
## Confusion Matrix and Statistics
##
##
## pred_rf1  No Yes
##      No   93  62
##      Yes 149 392
##
##                Accuracy : 0.6968
##                  95% CI : (0.6612, 0.7308)
##     No Information Rate : 0.6523
##     P-Value [Acc > NIR] : 0.007154
##
##                   Kappa : 0.2704
##
```

```
##  Mcnemar's Test P-Value : 3.21e-09
##
##             Sensitivity : 0.3843
##             Specificity : 0.8634
##          Pos Pred Value : 0.6000
##          Neg Pred Value : 0.7246
##              Prevalence : 0.3477
##          Detection Rate : 0.1336
##    Detection Prevalence : 0.2227
##       Balanced Accuracy : 0.6239
##
##        'Positive' Class : No
##
```

```
train_control <- trainControl(method = "cv", number = 5, search = "grid")
rf_grid <- expand.grid(.mtry = seq(1, 6, by = 1))
train_drafted <- subset(train, Drafted == "Yes")
optimized_rf2 <- randomForest(Pick ~ Sprint_40yd + Bench_Press_Reps + Broad_Jump + Vertical_Jump + Shut
print(optimized_rf2)
```

```
##
## Call:
##  randomForest(formula = Pick ~ Sprint_40yd + Bench_Press_Reps +      Broad_Jump + Vertical_Jump + Shu
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 5206.04
##                     % Var explained: -4.1
```

```
plot(optimized_rf2, main = "Optimized Random Forest")
```



```
# Assuming you already fitted the optimized_rf2 model using the given code
# Get the predicted values for the test set
predicted_values <- predict(optimized_rf2, newdata = test)

# Get the true values from the test set
true_values <- test$Pick

# Calculate the errors
errors <- predicted_values - true_values

# Calculate the Mean Absolute Error (MAE)
mae <- mean(abs(errors))
print(paste("MAE:", mae))
```

```
## [1] "MAE: 134.458115279436"
```

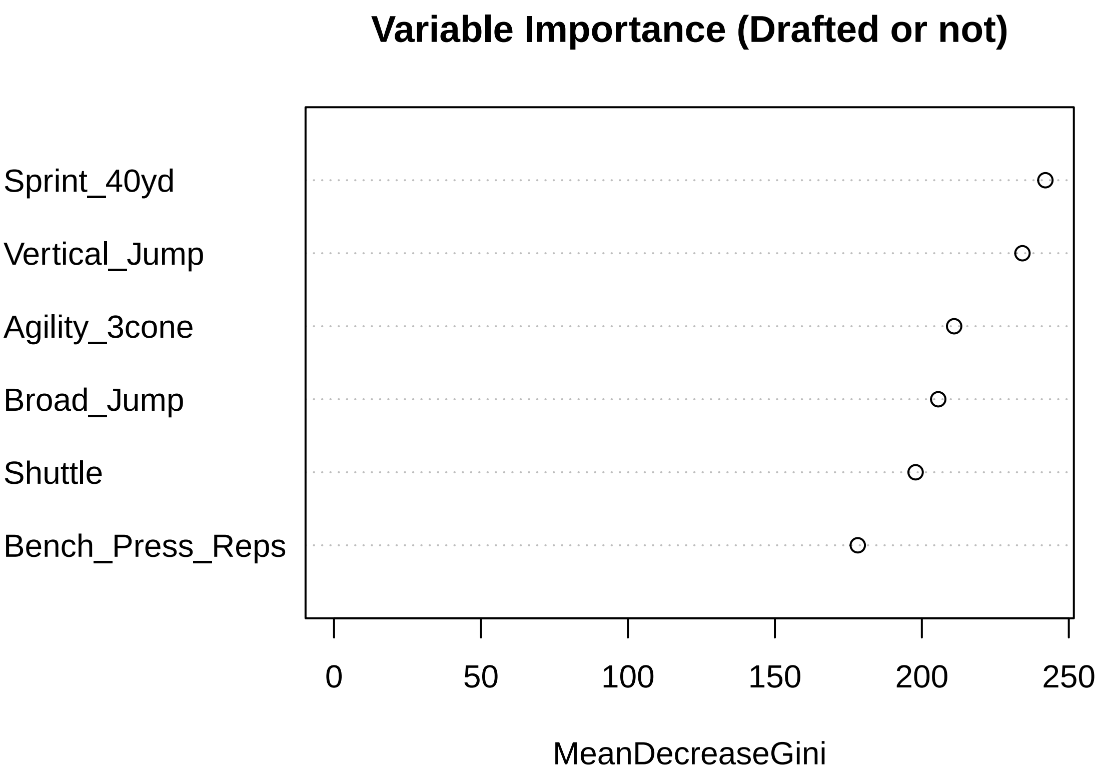
```
# Calculate the Mean Squared Error (MSE)
mse <- mean(errors^2)
print(paste("MSE:", mse))
```

```
## [1] "MSE: 29906.3854605501"
```

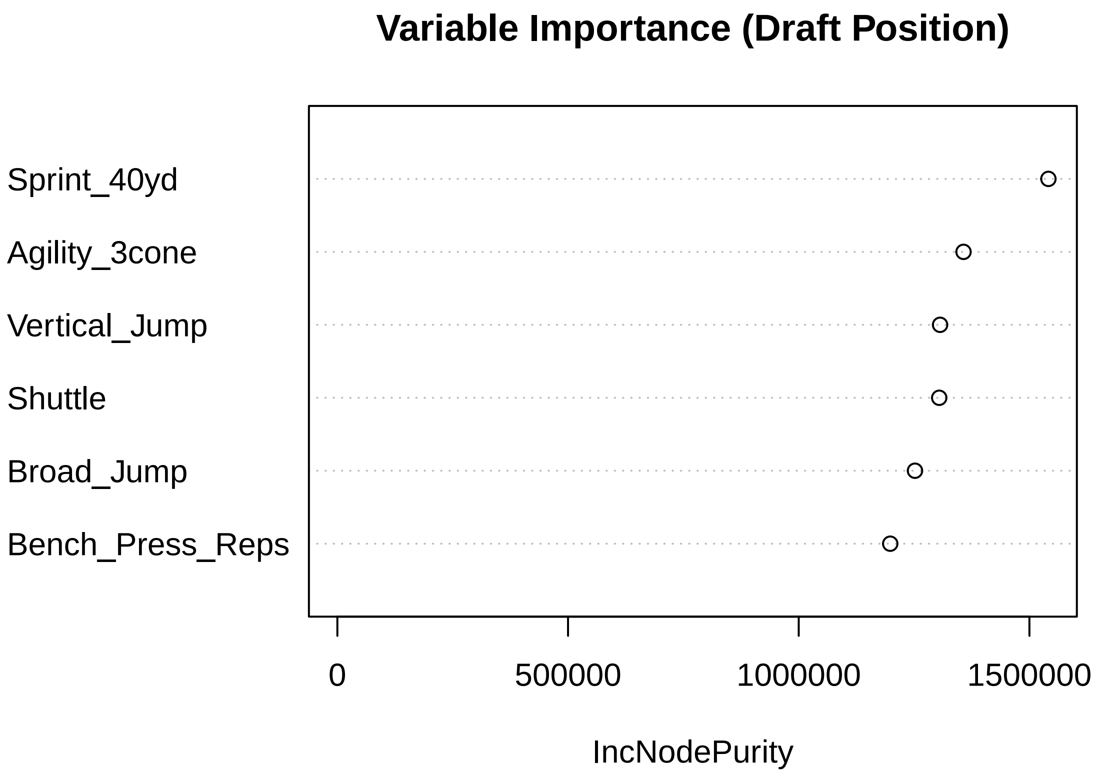
```
# Calculate the Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 172.934627708132"
```

```
varImpPlot(optimized_rf1, main = "Variable Importance (Drafted or not)")
```

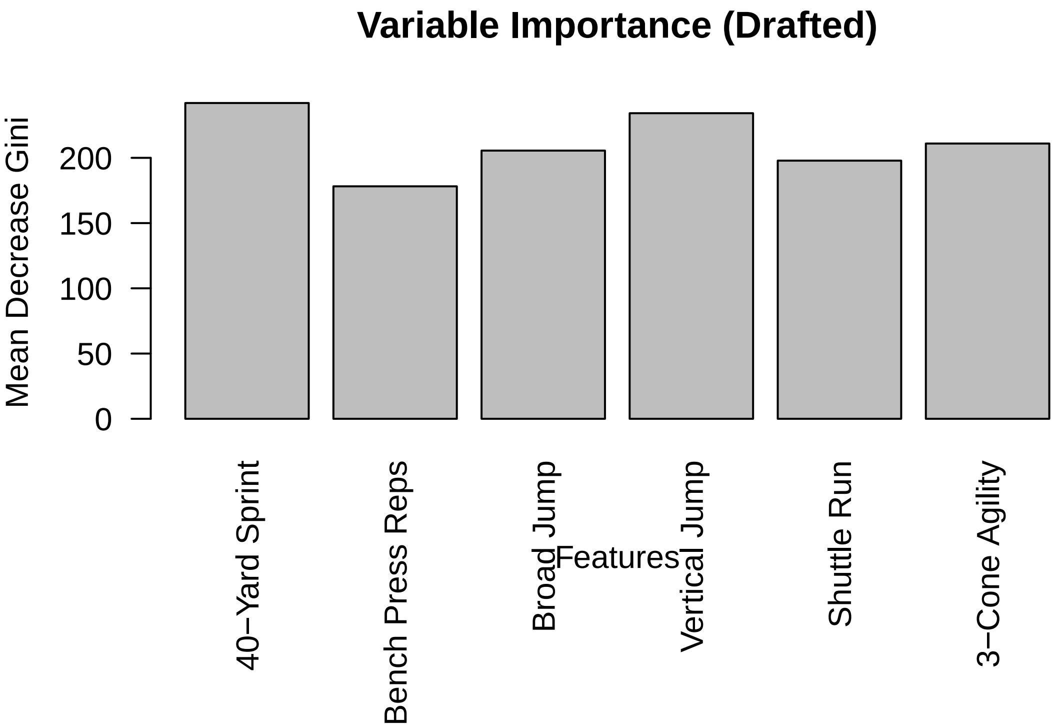


```
varImpPlot(optimized_rf2, main = "Variable Importance (Draft Position)")
```

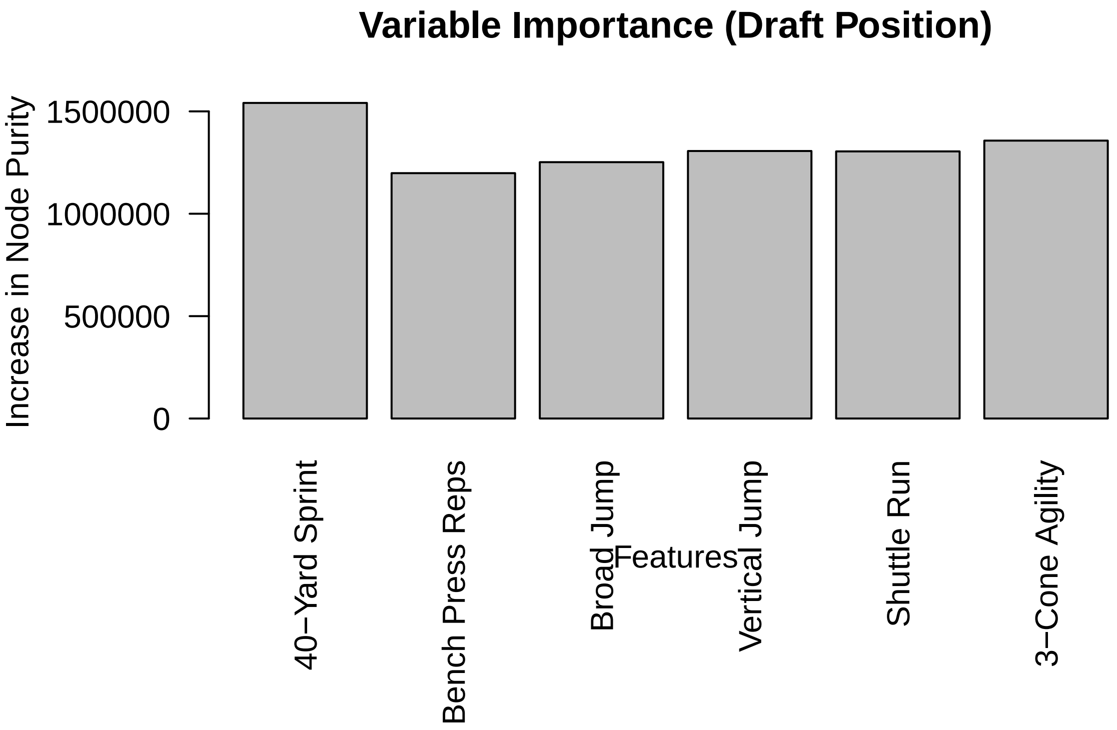


```
# Get the variable importance data
var_imp1 <- importance(optimized_rf1)
var_imp2 = importance(optimized_rf2)

# Create a named vector with the original variable names as names and the new names as values
new_var_names <- c(Sprint_40yd = "40-Yard Sprint",
                  Shuttle = "Shuttle Run",
                  Agility_3cone = "3-Cone Agility",
                  Broad_Jump = "Broad Jump",
                  Vertical_Jump = "Vertical Jump",
                  Bench_Press_Reps = "Bench Press Reps")

# Update the row names of the variable importance data
rownames(var_imp1) <- new_var_names[rownames(var_imp1)]
rownames(var_imp2) <- new_var_names[rownames(var_imp2)]
```

```
par(mar = c(10, 6, 4, 2) + 0.1)
# Create the variable importance plot with updated variable names
barplot(var_imp1[, "MeanDecreaseGini"], las = 2, main = "Variable Importance (Drafted)", xlab = "Featur
```



```
barplot(var_imp2[,"IncNodePurity"], las = 2, main = "Variable Importance (Draft Position)", xlab = "Fea
mtext("Increase in Node Purity", side = 2, line = 4.5)
```


Variable Importance (Draft Position)
Increase in Node Purity
1500000
1000000
500000
0
40−Yard Sprint
Bench Press Reps
Broad Jump
Vertical Jump
Shuttle Run
3−Cone Agility
Features