Franchise Investment Project

Daniel Jackson

February 4th, 2024

Packages Used

```
library(dplyr)
library(ggplot2)
library(scales)
library(corrplot)
library(glmnet)
library(pls)
library(tree)
library(randomForest)
```

Exploratory Analysis

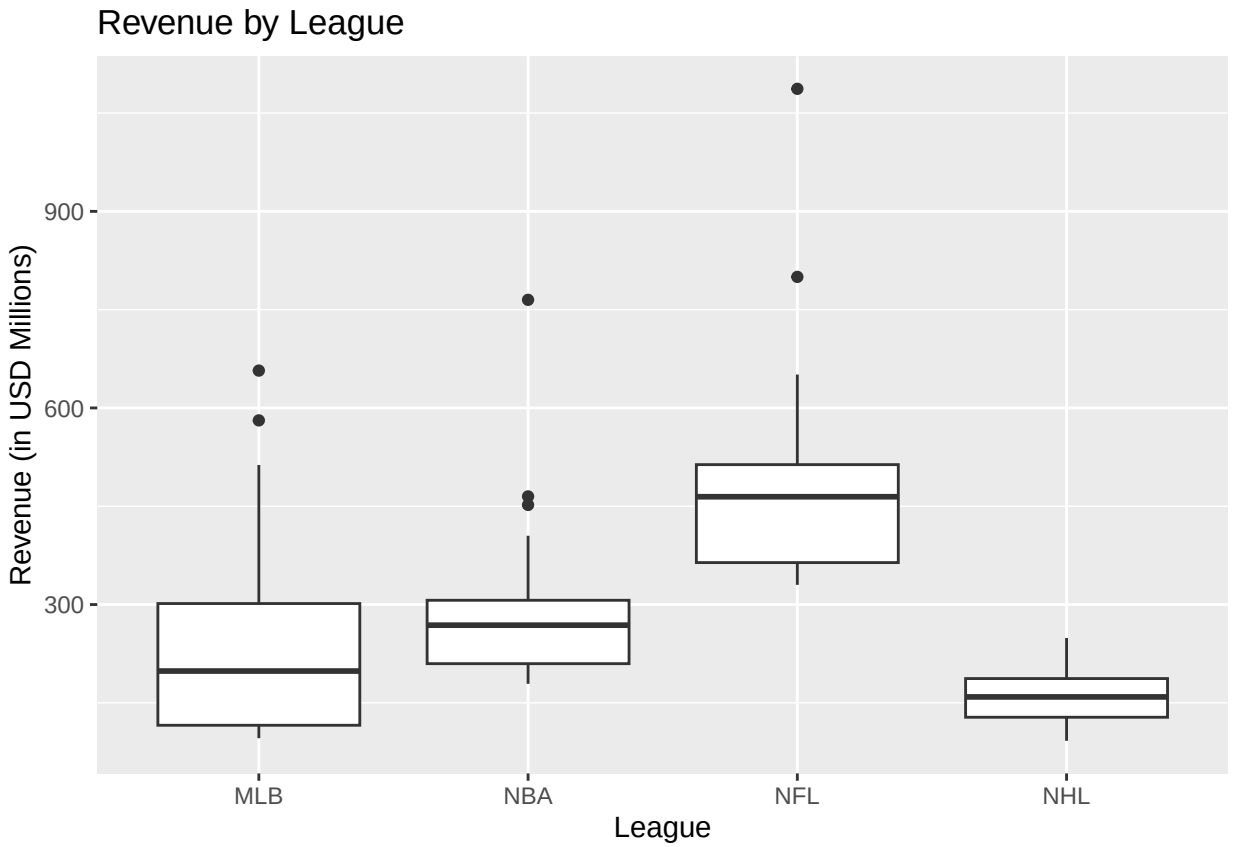
Let us look at the value-revenue ratio for each team in each league in data set. First we will create new predictor that divides the franchise's value by its revenue.

```
value_df$rev_to_val_ratio = value_df$revenue / value_df$value
# Look at average revenue to value ratio of all of the leagues
avg_rev_to_val_ratio = value_df %>%
  group_by(league) %>%
  summarise(average_rev_to_val = mean(rev_to_val_ratio, na.rm = TRUE))
print(avg_rev_to_val_ratio)
```

```
## # A tibble: 4 x 2
##   league average_rev_to_val
##   <chr>               <dbl>
## 1 MLB                 0.120
## 2 NBA                 0.112
## 3 NFL                 0.118
## 4 NHL                 0.216
```

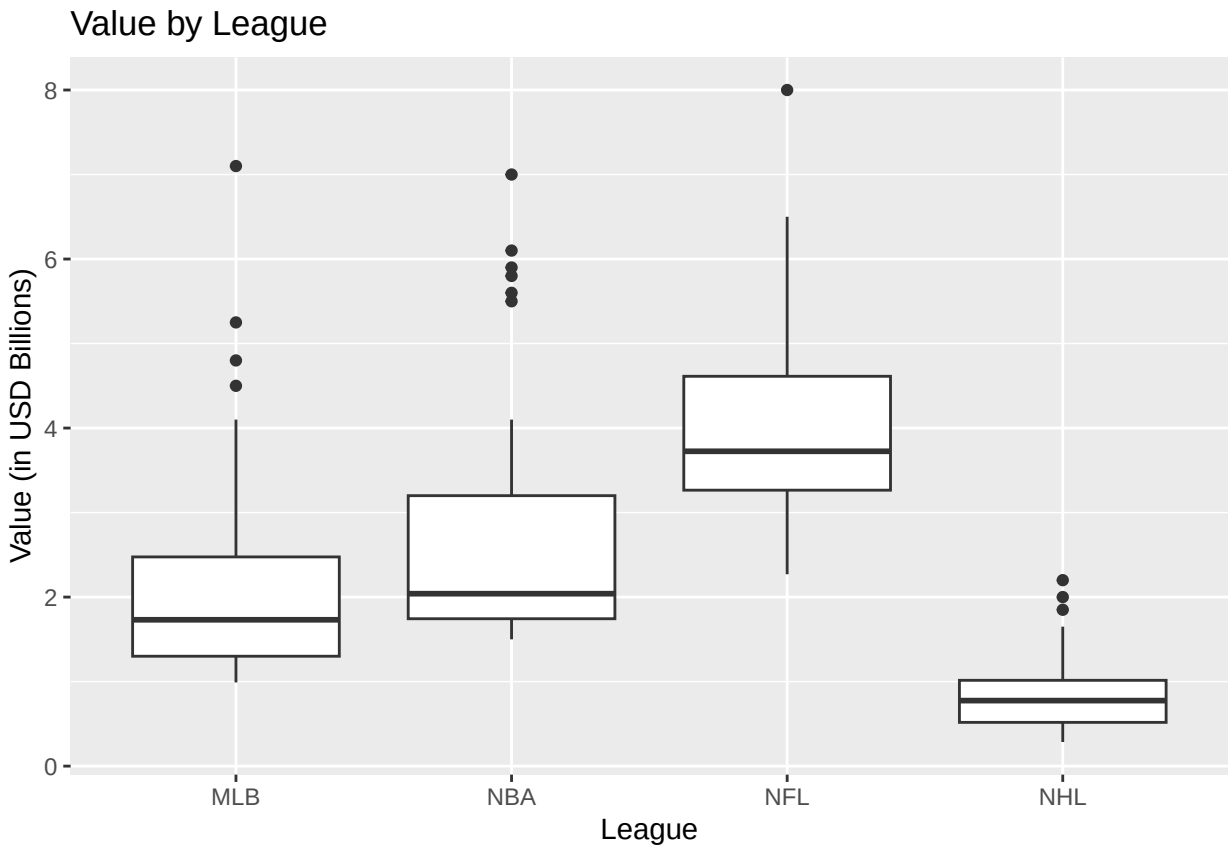
The NHL, on average, has the highest revenue to value ratio of all the leagues.
Let's look at box plots for each predictor by league, just to get an idea of how the data is distributed.
Let's look at the distribution of revenue between leagues.

```
selected_predictors = c("revenue", "value", "income", "population",
                        "rev_to_value_ratio")
# Create box plots using ggplot2
# Revenue
ggplot(value_df, aes(x = league, y = .data[[selected_predictors[1]]]/1e6)) +
  geom_boxplot() +
  labs(title = paste("Revenue by League"),
       x = "League",
       y = "Revenue (in USD Millions)")
```



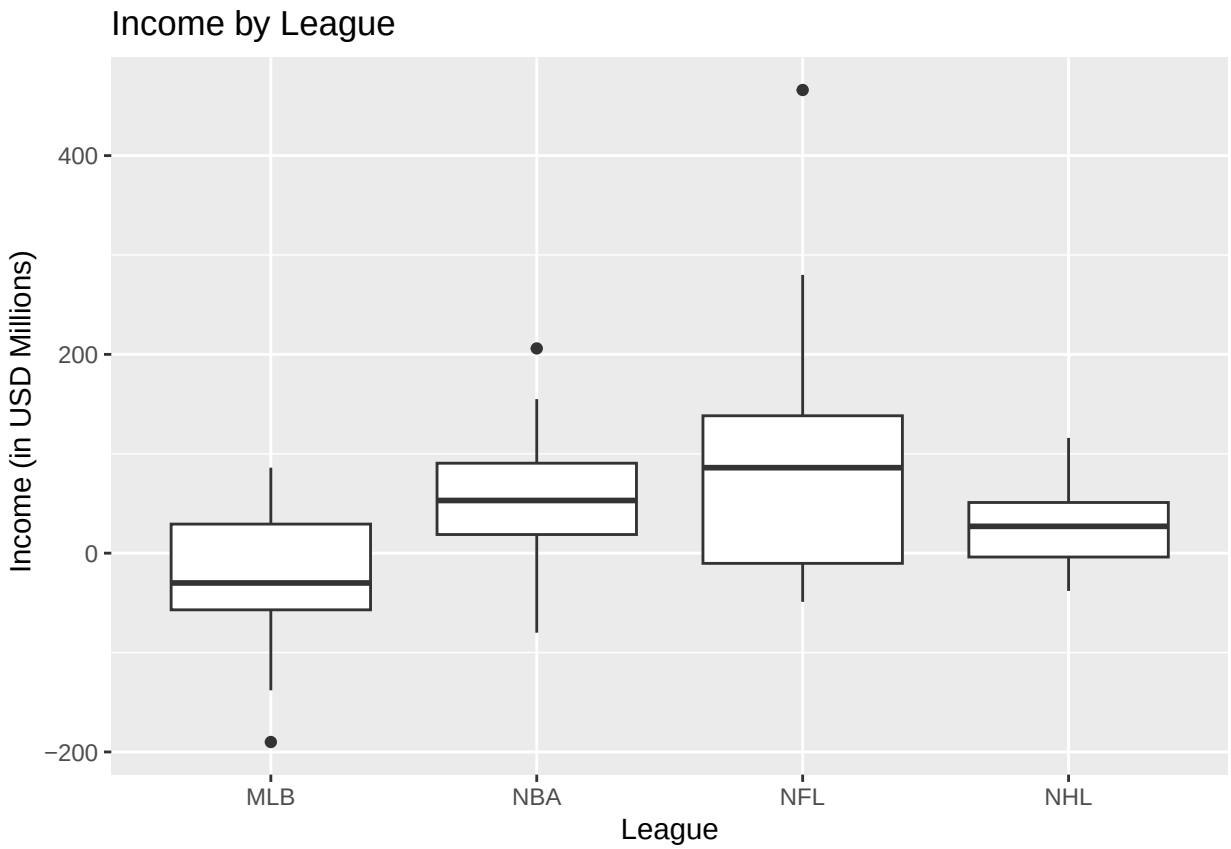
Now, let's look at value.

```
ggplot(value_df, aes(x = league, y = .data[[selected_predictors[2]]]/1e9)) +
  geom_boxplot() +
  labs(title = paste("Value by League"),
       x = "League",
       y = "Value (in USD Billions)")
```



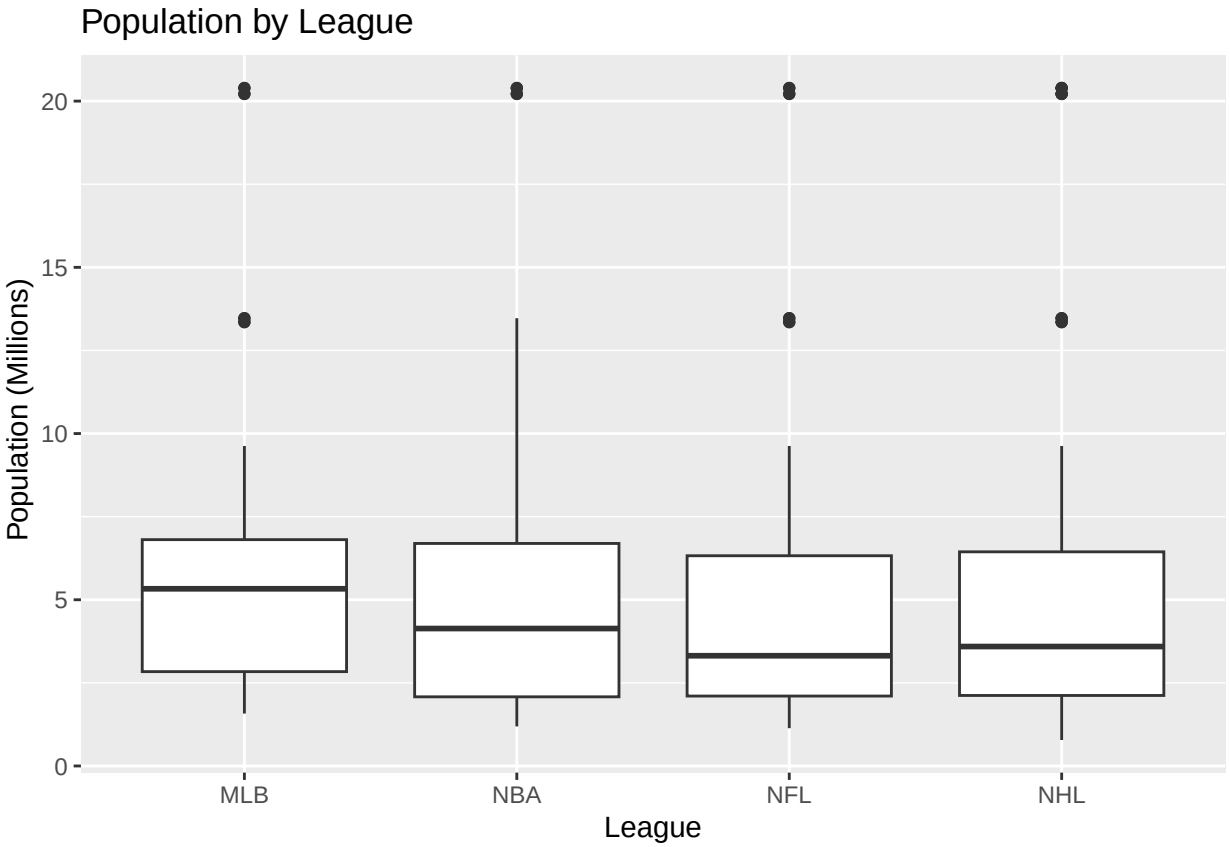
Now, let's look at income.

```
ggplot(value_df, aes(x = league, y = .data[[selected_predictors[3]]]/1e6)) +
  geom_boxplot() +
  labs(title = paste("Income by League"),
       x = "League",
       y = " Income (in USD Millions)")
```



And finally, by population.

```
ggplot(value_df, aes(x = league, y = .data[[selected_predictors[4]]]/1e6)) +
  geom_boxplot() +
  labs(title = paste("Population by League"),
       x = "League",
       y = " Population (Millions)")
```



Now, looking at big picture, let's look at and average revenue and value for each of the major sports in both 2021 and 2023. Let's then graph it.

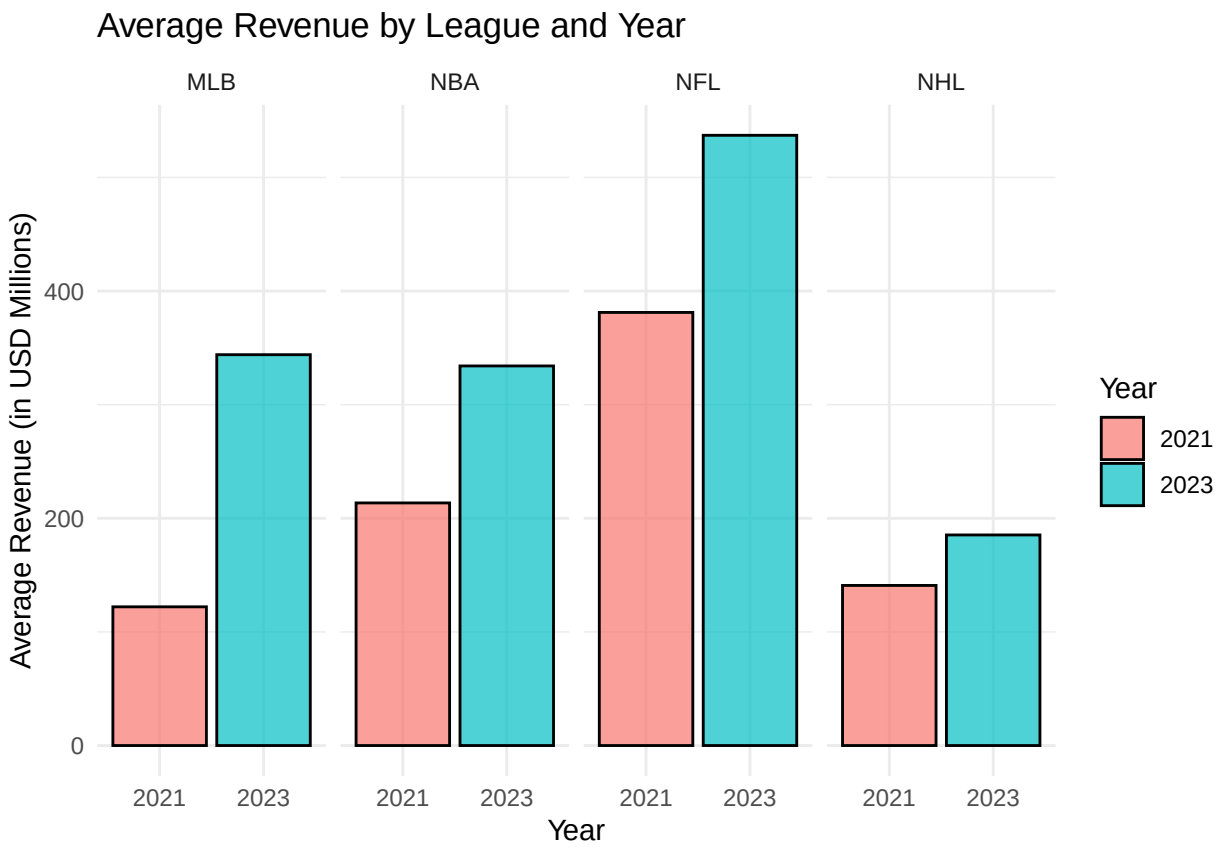
```
average_revenue = value_df %>%
  group_by(year, league) %>%
  summarise(avg_revenue = mean(revenue, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
print(average_revenue)
```

```
## # A tibble: 8 x 3
## # Groups:   year [2]
##    year league avg_revenue
##   <int> <chr>        <dbl>
## 1  2021 MLB     122100000
## 2  2021 NBA     213533333.
## 3  2021 NFL     381187500
## 4  2021 NHL     140903226.
## 5  2023 MLB     344000000
## 6  2023 NBA     334100000
## 7  2023 NFL     537093750
## 8  2023 NHL     185343750
```

```
# Plot it
ggplot(average_revenue, aes(x = as.factor(year), y = avg_revenue / 1e6, fill = as.factor(year))) +
  geom_bar(stat = "identity", position = "dodge", color = "black", alpha = 0.7) +
  facet_grid(. ~ league) +
  labs(title = "Average Revenue by League and Year",
       x = "Year",
       y = "Average Revenue (in USD Millions)",
       fill = "Year") +
  scale_y_continuous(labels = scales::comma) +  # Use comma as the thousand separator
  theme_minimal()
```



And now for the value.

```
average_value = value_df %>%
  group_by(year, league) %>%
  summarise(avg_value = mean(value, na.rm = TRUE))
```

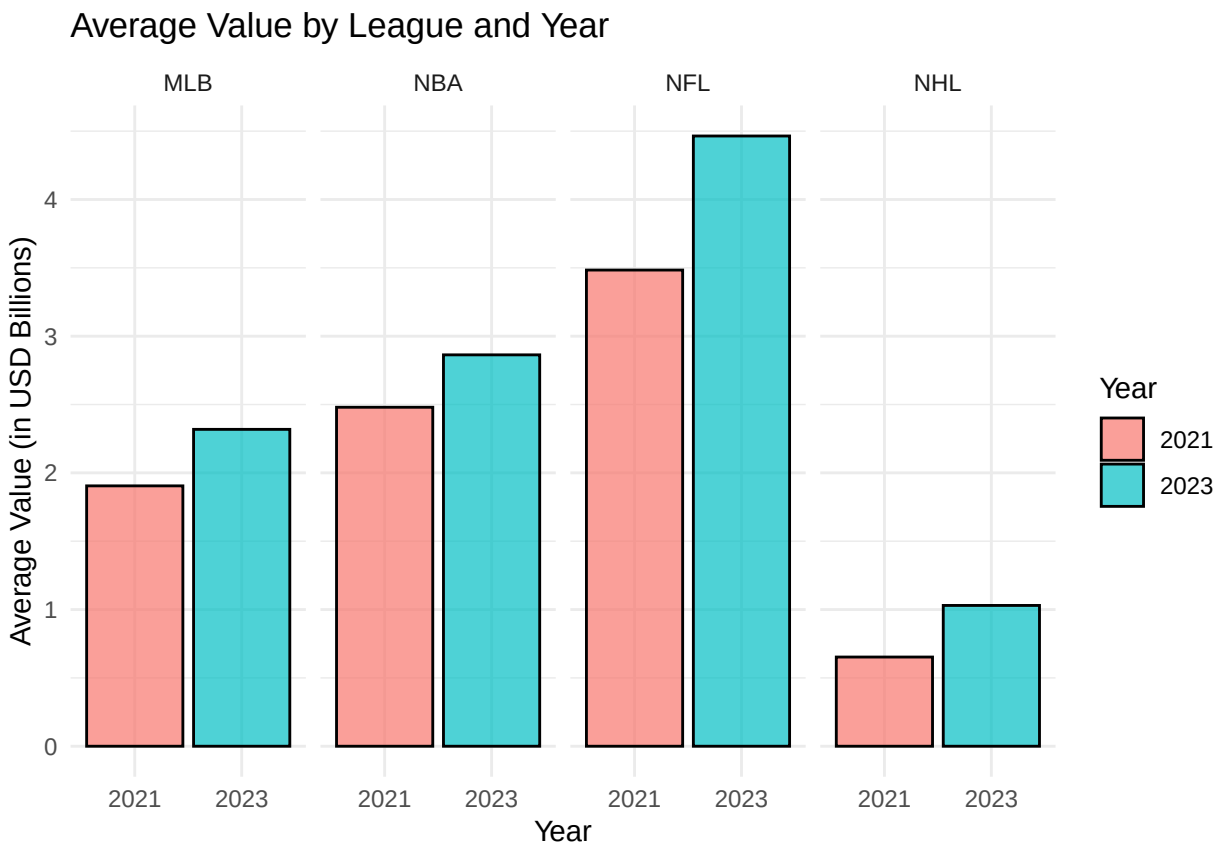
```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
print(average_revenue)
```

```
## # A tibble: 8 x 3
## # Groups:   year [2]
##    year league avg_revenue
```

```
##   <int> <chr>        <dbl>
## 1  2021 MLB     122100000
## 2  2021 NBA     213533333.
## 3  2021 NFL     381187500
## 4  2021 NHL     140903226.
## 5  2023 MLB     344000000
## 6  2023 NBA     334100000
## 7  2023 NFL     537093750
## 8  2023 NHL     185343750
```

```r
# Plot it
ggplot(average_value, aes(x = as.factor(year), y = avg_value / 1e9, fill = as.factor(year))) +
  geom_bar(stat = "identity", position = "dodge", color = "black", alpha = 0.7) +
  facet_grid(. ~ league) +
  labs(title = "Average Value by League and Year",
       x = "Year",
       y = "Average Value (in USD Billions)",
       fill = "Year") +
  scale_y_continuous(labels = scales::comma) +  # Use comma as the thousand separator
  theme_minimal()
```



Now, let's look at the top 5 teams from 2021 and 2023 with the highest revenues.
For 2021:

```r
filtered_df = value_df %>%
  filter(year == 2021)
sorted_df = filtered_df %>%
```

```
  arrange(desc(revenue))
top_5_teams = head(sorted_df, 5)
print(top_5_teams)
```

```
##   year league                team  revenue     value yoy_val_inc debt_value
## 1 2021    NFL       Dallas Cowboys 8.00e+08 6.500e+09        0.14       0.05
## 2 2021    NFL New England Patriots 4.78e+08 5.000e+09        0.14       0.04
## 3 2021    NFL     Los Angeles Rams 4.22e+08 4.800e+09        0.20       0.66
## 4 2021    NFL       Houston Texans 3.97e+08 3.700e+09        0.12       0.00
## 5 2021    NFL    Las Vegas Raiders 3.89e+08 3.415e+09        0.10       0.45
##      income population rev_to_val_ratio
## 1 280000000    7255746       0.12307692
## 2 142000000    4809310       0.09560000
## 3  37000000   13351709       0.08791667
## 4 -20000000    6793465       0.10729730
## 5  -5100000    2162347       0.11390922
```

For 2023:

```
filtered_df = value_df %>%
  filter(year == 2023)
sorted_df = filtered_df %>%
  arrange(desc(revenue))
top_5_teams = head(sorted_df, 5)
print(top_5_teams)
```

```
##   year league                  team   revenue   value yoy_val_inc debt_value
## 1 2023    NFL        Dallas Cowboys 1.087e+09 8.0e+09        0.23       0.04
## 2 2023    NBA Golden State Warriors 7.650e+08 7.0e+09        0.25       0.16
## 3 2023    MLB      New York Yankees 6.570e+08 7.1e+09        0.18       0.00
## 4 2023    NFL  New England Patriots 6.510e+08 6.4e+09        0.28       0.03
## 5 2023    NFL      Los Angeles Rams 6.280e+08 6.2e+09        0.29       0.51
##     income population rev_to_val_ratio
## 1 4.66e+08    7319735       0.13587500
## 2 2.06e+08    6737521       0.10928571
## 3 1.60e+07   20394396       0.09253521
## 4 2.31e+08    4851723       0.10171875
## 5 2.03e+08   13469458       0.10129032
```

Let's look at the top 5 teams from 2021 and 2023 with the highest values. For 2021:

```
filtered_df = value_df %>%
  filter(year == 2021)
sorted_df = filtered_df %>%
  arrange(desc(value))
top_5_teams = head(sorted_df, 5)
print(top_5_teams)
```

```
##   year league                  team  revenue    value yoy_val_inc debt_value
## 1 2021    NFL        Dallas Cowboys 8.00e+08 6.50e+09        0.14       0.05
## 2 2021    NBA       New York Knicks 2.98e+08 5.80e+09        0.16       0.07
```

```
## 3 2021    NBA Golden State Warriors 2.58e+08 5.60e+09        0.19       0.20
## 4 2021    NBA    Los Angeles Lakers 3.16e+08 5.50e+09        0.20       0.02
## 5 2021    MLB      New York Yankees 1.08e+08 5.25e+09        0.05       0.00
##     income population rev_to_val_ratio
## 1  2.8e+08    7255746       0.12307692
## 2  7.1e+07   20216110       0.05137931
## 3 -4.4e+07    6678622       0.04607143
## 4  6.3e+07   13351709       0.05745455
## 5 -1.9e+08   20216110       0.02057143
```

For 2023:

```
filtered_df = value_df %>%
  filter(year == 2023)
sorted_df = filtered_df %>%
  arrange(desc(value))
top_5_teams = head(sorted_df, 5)
print(top_5_teams)
```

```
##   year league                  team   revenue   value yoy_val_inc debt_value
## 1 2023    NFL        Dallas Cowboys 1.087e+09 8.0e+09        0.23       0.04
## 2 2023    MLB      New York Yankees 6.570e+08 7.1e+09        0.18       0.00
## 3 2023    NBA Golden State Warriors 7.650e+08 7.0e+09        0.25       0.16
## 4 2023    NFL  New England Patriots 6.510e+08 6.4e+09        0.28       0.03
## 5 2023    NFL      Los Angeles Rams 6.280e+08 6.2e+09        0.29       0.51
##     income population rev_to_val_ratio
## 1 4.66e+08    7319735       0.13587500
## 2 1.60e+07   20394396       0.09253521
## 3 2.06e+08    6737521       0.10928571
## 4 2.31e+08    4851723       0.10171875
## 5 2.03e+08   13469458       0.10129032
```

Overall, Cowboys are highest valued team in our data set. They also have the highest revenue in 2021 and 2023, so it makes sense that they are the highest valued team.

Let's look at highest average year over year value increase for each league for 2021 and 2023.
For 2021:

```
filtered_df = value_df %>%
  filter(year == 2021)
avg_yoy_increase = filtered_df %>%
  group_by(league) %>%
  summarise(avg_yoy_increase = mean(yoy_val_inc, na.rm = TRUE))
print(avg_yoy_increase)
```

```
## # A tibble: 4 x 2
##   league avg_yoy_increase
##   <chr>             <dbl>
## 1 MLB              0.0233
## 2 NBA              0.113
## 3 NFL              0.145
## 4 NHL             -0.0271
```

On average, NFL has biggest year over year value increase.
For 2023:

```
filtered_df = value_df %>%
  filter(year == 2023)
avg_yoy_increase = filtered_df %>%
  group_by(league) %>%
  summarise(avg_yoy_increase = mean(yoy_val_inc, na.rm = TRUE))
print(avg_yoy_increase)
```

```
## # A tibble: 4 x 2
##   league avg_yoy_increase
##   <chr>             <dbl>
## 1 MLB               0.106
## 2 NBA               0.158
## 3 NFL               0.286
## 4 NHL               0.218
```

The NHL has biggest year over year value increase.
The NHL went from being last in 2021 to second behind NFL in 2023. Hockey seems to be having success as a league if average year over year value increase is greater than both NBA and MLB.

Let's look at average debt value for each league for 2021 and 2023.
For 2021:

```
filtered_df = value_df %>%
  filter(year == 2021)
avg_debt_value = filtered_df %>%
  group_by(league) %>%
  summarise(debt_value = mean(debt_value, na.rm = TRUE))
print(avg_debt_value)
```

```
## # A tibble: 4 x 2
##   league debt_value
##   <chr>       <dbl>
## 1 MLB         0.134
## 2 NBA         0.120
## 3 NFL         0.122
## 4 NHL         0.276
```

The NBA has lowest debt value of all the leagues in 2021.
For 2023:

```
filtered_df = value_df %>%
  filter(year == 2023)
avg_debt_value = filtered_df %>%
  group_by(league) %>%
  summarise(debt_value = mean(debt_value, na.rm = TRUE))
print(avg_debt_value)
```

```
## # A tibble: 4 x 2
##   league debt_value
```

```
##   <chr>       <dbl>
## 1 MLB        0.120
## 2 NBA        0.104
## 3 NFL        0.0931
## 4 NHL        0.191
```

In 2023, NFL has lowest debt value of all four sports.
Let's look at average income for each league for 2021 and 2023.
For 2021:

```
filtered_df = value_df %>%
  filter(year == 2021)
avg_income = filtered_df %>%
  group_by(league) %>%
  summarise(avgerage_income = mean(income, na.rm = TRUE))
print(avg_income)
```

```
## # A tibble: 4 x 2
##   league avgerage_income
##   <chr>            <dbl>
## 1 MLB         -60066667.
## 2 NBA          24606667.
## 3 NFL            6990625
## 4 NHL            7901983.
```

The NFL with the highest average income value in 2021.
For 2023:

```
filtered_df = value_df %>%
  filter(year == 2023)
avg_income = filtered_df %>%
  group_by(league) %>%
  summarise(average_income = mean(income, na.rm = TRUE))
print(avg_income)
```

```
## # A tibble: 4 x 2
##   league average_income
##   <chr>           <dbl>
## 1 MLB          17653333.
## 2 NBA          90800000
## 3 NFL         145968750
## 4 NHL          47730046.
```

Once again, NFL with highest average income value in 2023. Based on our exploratory analysis, the NFL seems to be the most profitable league. Now let's turn our focus to some regression models that will help us predict future franchise's values using the data that we have.

Using Regression to Predict Franchise's Values

Let's look at correlation between numeric values in dataset.

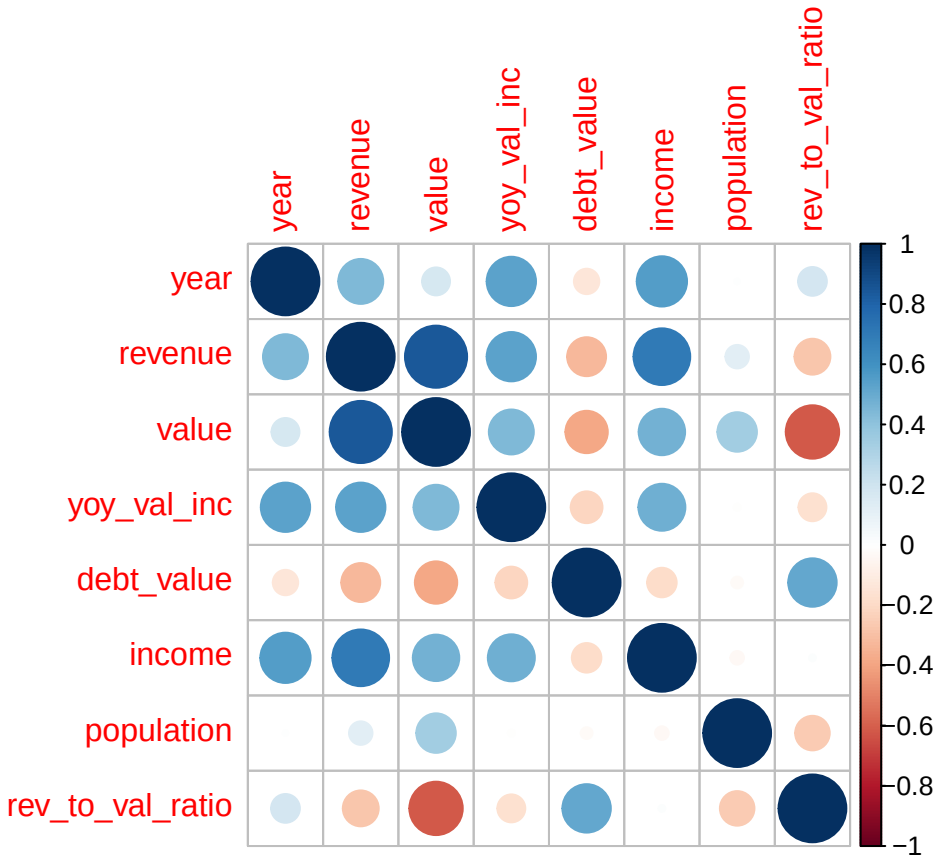
```r
cor_df = value_df %>%
  select_if(is.numeric)
cor(cor_df)
```

```
##                          year    revenue      value  yoy_val_inc  debt_value
## year              1.000000000  0.4405609  0.1705285  0.534354524 -0.13985078
## revenue           0.440560868  1.0000000  0.8426507  0.528394893 -0.33286734
## value             0.170528515  0.8426507  1.0000000  0.435954815 -0.39461338
## yoy_val_inc       0.534354524  0.5283949  0.4359548  1.000000000 -0.22199962
## debt_value       -0.139850781 -0.3328673 -0.3946134 -0.221999618  1.00000000
## income            0.552803645  0.7036908  0.4748500  0.475031652 -0.19256182
## population        0.003400134  0.1175508  0.3443548 -0.008885416 -0.03453062
## rev_to_val_ratio  0.177518028 -0.2760820 -0.6234149 -0.173552418  0.51117375
##                      income   population rev_to_val_ratio
## year              0.55280365  0.003400134        0.1775180
## revenue           0.70369076  0.117550760       -0.2760820
## value             0.47485003  0.344354821       -0.6234149
## yoy_val_inc       0.47503165 -0.008885416       -0.1735524
## debt_value       -0.19256182 -0.034530615        0.5111737
## income            1.00000000 -0.044277614        0.0138141
## population       -0.04427761  1.000000000       -0.2617921
## rev_to_val_ratio  0.01381410 -0.261792114        1.0000000
```

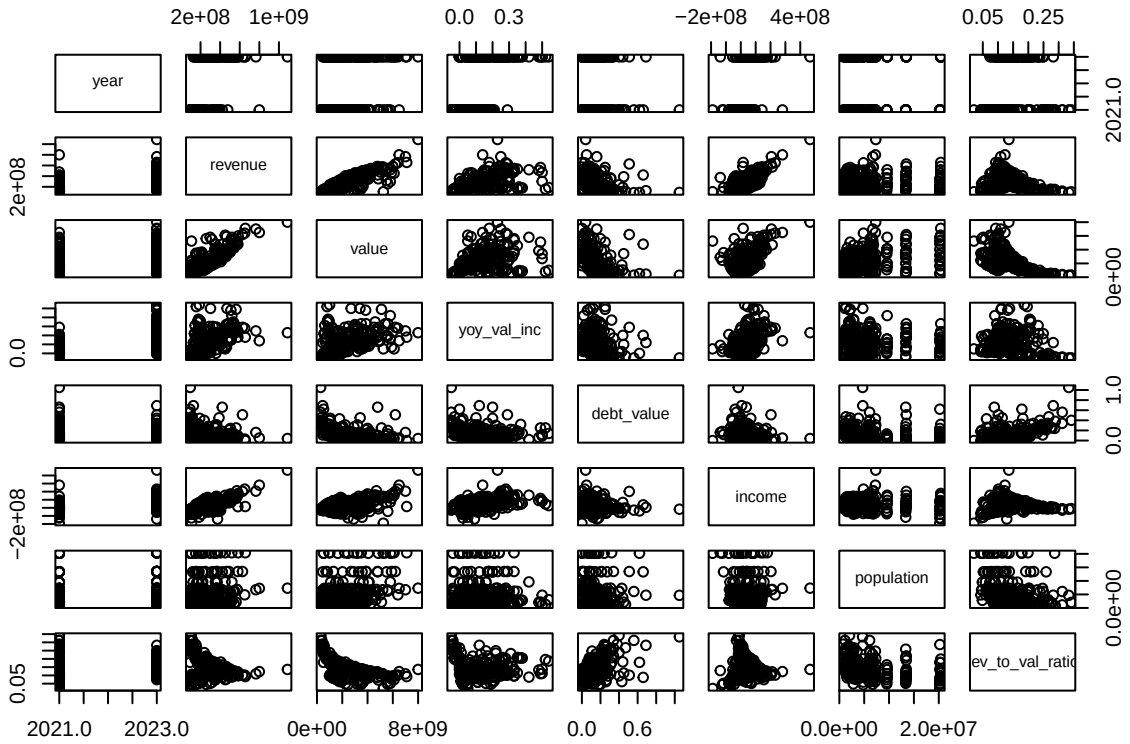
```r
c = round(cor(cor_df), digits = 2)
cor(cor_df)
```

```
##                          year    revenue      value  yoy_val_inc  debt_value
## year              1.000000000  0.4405609  0.1705285  0.534354524 -0.13985078
## revenue           0.440560868  1.0000000  0.8426507  0.528394893 -0.33286734
## value             0.170528515  0.8426507  1.0000000  0.435954815 -0.39461338
## yoy_val_inc       0.534354524  0.5283949  0.4359548  1.000000000 -0.22199962
## debt_value       -0.139850781 -0.3328673 -0.3946134 -0.221999618  1.00000000
## income            0.552803645  0.7036908  0.4748500  0.475031652 -0.19256182
## population        0.003400134  0.1175508  0.3443548 -0.008885416 -0.03453062
## rev_to_val_ratio  0.177518028 -0.2760820 -0.6234149 -0.173552418  0.51117375
##                      income   population rev_to_val_ratio
## year              0.55280365  0.003400134        0.1775180
## revenue           0.70369076  0.117550760       -0.2760820
## value             0.47485003  0.344354821       -0.6234149
## yoy_val_inc       0.47503165 -0.008885416       -0.1735524
## debt_value       -0.19256182 -0.034530615        0.5111737
## income            1.00000000 -0.044277614        0.0138141
## population       -0.04427761  1.000000000       -0.2617921
## rev_to_val_ratio  0.01381410 -0.261792114        1.0000000
```

```r
# Correlation plot
corrplot::corrplot(c)
```

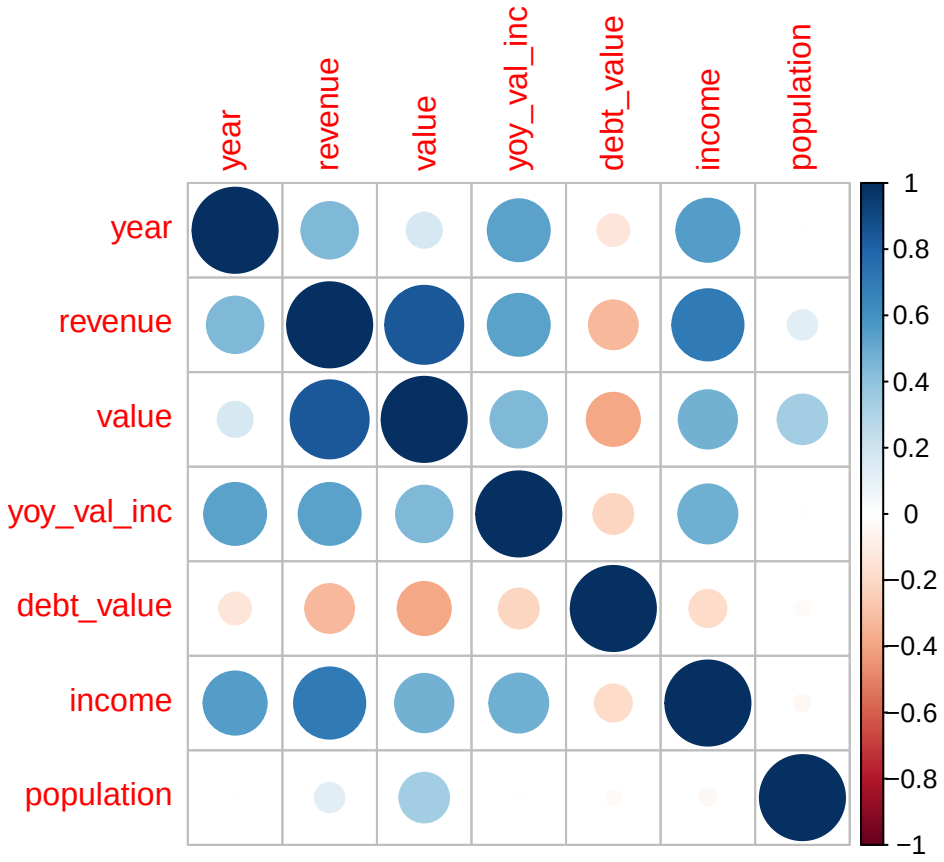



```
pairs(cor_df)
```



Let's remove rev_to_val_ratio as it is could create mutlicolliearity issues:

```
cor_df = subset(cor_df, select = -rev_to_val_ratio)
c = round(cor(cor_df), digits = 2)
corrplot::corrplot(c)
```



Now, let's create our training and test data sets so we can test our trained regression models to find mean squared errors so we an analyze how accurate our models really are.

```
dim(value_df)
```

```
## [1] 247  10
```

```
247 / 2
```

```
## [1] 123.5
```

```
set.seed(1)
train = sample(1:nrow(cor_df), 123)
value_train = cor_df[train,]
value_test = cor_df[-train,]
```

Try linear regression to predict value in data set.

```
value_lm = lm(value ~., value_train)
summary(value_lm)
```

```
##
## Call:
## lm(formula = value ~ ., data = value_train)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.526e+09 -3.955e+08 -3.762e+07  2.423e+08  2.926e+09
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.853e+11  1.463e+11   6.052 1.80e-08 ***
## year        -4.382e+08  7.237e+07  -6.055 1.77e-08 ***
## revenue      9.617e+00  5.573e-01  17.257  < 2e-16 ***
## yoy_val_inc  1.436e+09  5.639e+08   2.546   0.0122 *
## debt_value  -1.194e+09  4.684e+08  -2.550   0.0121 *
## income      -4.255e-01  1.279e+00  -0.333   0.7400
## population   5.910e+01  1.098e+01   5.382 3.88e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 626800000 on 116 degrees of freedom
## Multiple R-squared:  0.8443, Adjusted R-squared:  0.8362
## F-statistic: 104.8 on 6 and 116 DF,  p-value: < 2.2e-16
```

Using null hypothesis testing, we see that all predictors pass null hypothesis test, since p-values are less than 0.05.
The model has a R^2 value of 84.15.
Let's use trained model to predict test data.

```
pred_value_lm = predict(value_lm, newdata = value_test)
mean((pred_value_lm - value_test$value)^2)
```

```
## [1] 4.491086e+17
```

Our model produced a very high mean squared error value due to high value of the response variable. In order to work around this high mean squared error rate, let's log transform our repsponse variable and refit our linear model.

```
value_train$log_value = log(value_train$value)
value_test$log_value = log(value_test$value)
log_value_lm = lm(log_value ~. - value, value_train)
summary(log_value_lm)
```

```
##
## Call:
## lm(formula = log_value ~ . - value, data = value_train)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.89167 -0.20541  0.00541  0.18494  0.94889
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.636e+02  7.665e+01   4.744 6.02e-06 ***
## year        -1.698e-01  3.792e-02  -4.479 1.77e-05 ***
## revenue      4.221e-09  2.920e-10  14.457  < 2e-16 ***
## yoy_val_inc  6.594e-01  2.955e-01   2.232  0.02757 *
## debt_value  -1.514e+00  2.454e-01  -6.169 1.03e-08 ***
## income      -1.320e-09  6.703e-10  -1.969  0.05131 .
## population   1.522e-08  5.753e-09   2.646  0.00927 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3284 on 116 degrees of freedom
## Multiple R-squared:  0.799,  Adjusted R-squared:  0.7887
## F-statistic: 76.87 on 6 and 116 DF,  p-value: < 2.2e-16
```

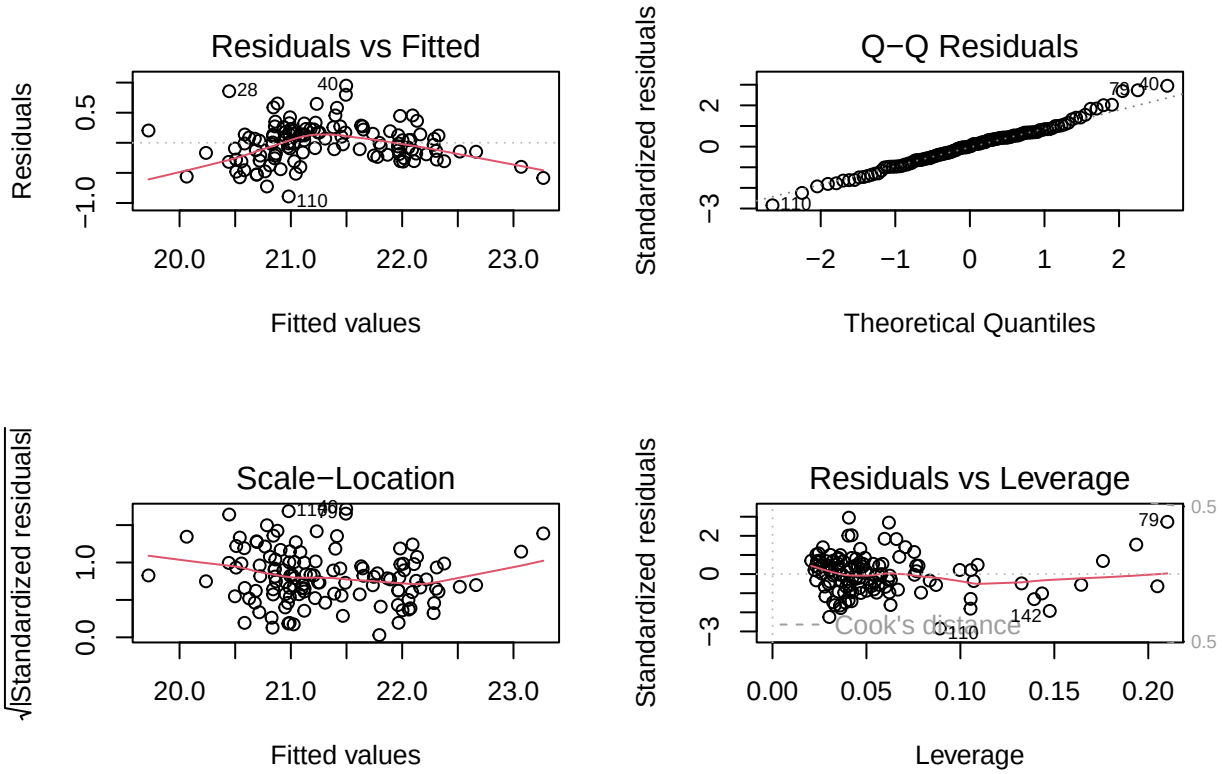
All predictors pass null hypothesis test.
R^2 value of 0.79.
Let's make predictions on our trained model.

```
pred_value_lm = predict(log_value_lm, newdata = value_test)
mean((pred_value_lm - value_test$log_value)^2)
```

```
## [1] 0.1388793
```

This model produced a mean squqred error of 0.139 after we log transforemd our value resonse variable. We will keep that transformation on the value variable throughout the rest of our regression analysis.
Let's plot our linear model.

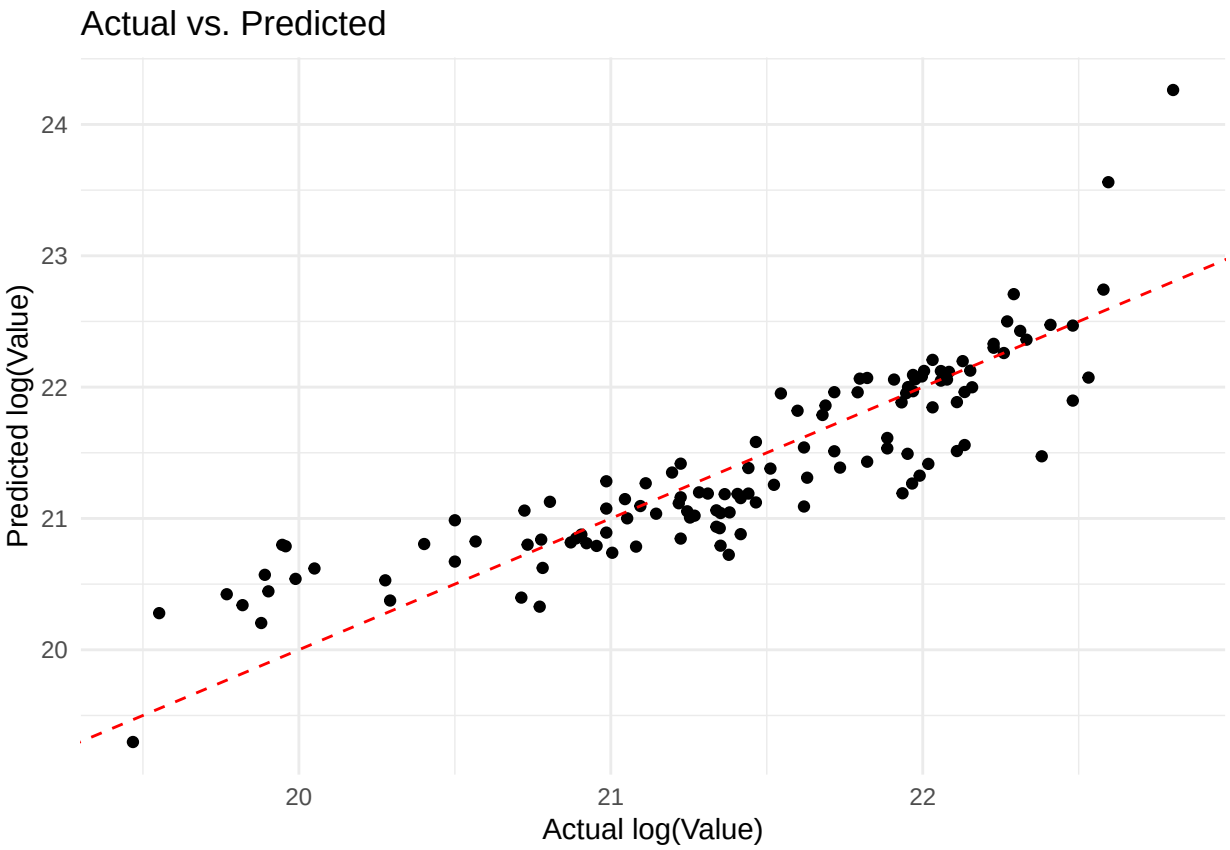
```
par(mfrow = c(2, 2))
plot(log_value_lm)
```



Let's also plot the actual values versus the predicted values.

```
par(mfrow = c(1, 1))
plot_df = data.frame(Actual = value_test$log_value, Predicted = pred_value_lm)

ggplot(plot_df, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  # Adds a line of perfect prediction
  labs(x = "Actual log(Value)",
       y = "Predicted log(Value)",
       title = "Actual vs. Predicted") +
  theme_minimal()
```



Let's try some transformations on the predictors besides year.
Let's start with a square root transformation.

```
sqrt_value_lm = lm(log_value ~ - value + year + sqrt(revenue) +
                     sqrt(yoy_val_inc) + sqrt(debt_value) +
                     sqrt(income) + sqrt(population), value_train)
```

```
## Warning in sqrt(yoy_val_inc): NaNs produced
```

```
## Warning in sqrt(income): NaNs produced
```

```
summary(sqrt_value_lm)
```

```
##
## Call:
## lm(formula = log_value ~ -value + year + sqrt(revenue) + sqrt(yoy_val_inc) +
##     sqrt(debt_value) + sqrt(income) + sqrt(population), data = value_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36777 -0.10586 -0.01422  0.11216  0.44740
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        4.945e+02  5.164e+01   9.576 1.56e-14 ***
```

```
## year              -2.352e-01  2.555e-02  -9.205 7.68e-14 ***
## sqrt(revenue)      1.418e-04  6.689e-06  21.198  < 2e-16 ***
## sqrt(yoy_val_inc)  7.900e-02  1.312e-01   0.602 0.548985
## sqrt(debt_value)  -7.558e-01  1.475e-01  -5.124 2.36e-06 ***
## sqrt(income)       2.138e-05  8.701e-06   2.457 0.016367 *
## sqrt(population)   8.716e-05  2.138e-05   4.077 0.000115 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1728 on 73 degrees of freedom
##   (43 observations deleted due to missingness)
## Multiple R-squared:  0.9381, Adjusted R-squared:  0.933
## F-statistic: 184.4 on 6 and 73 DF,  p-value: < 2.2e-16
```

We see that yoy_val_inc and income have p-values greater than 0.05 so we will remove them from the model.

```
sqrt_value_lm = lm(log_value ~ - value + year + sqrt(revenue) +
                     sqrt(debt_value) + sqrt(population), value_train)
summary(sqrt_value_lm)
```

```
##
## Call:
## lm(formula = log_value ~ -value + year + sqrt(revenue) + sqrt(debt_value) +
##     sqrt(population), data = value_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91198 -0.14054  0.00013  0.14490  1.02555
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.669e+02  6.390e+01   5.741 7.45e-08 ***
## year             -1.719e-01  3.163e-02  -5.435 2.99e-07 ***
## sqrt(revenue)     1.454e-04  8.317e-06  17.484  < 2e-16 ***
## sqrt(debt_value) -1.227e+00  2.048e-01  -5.990 2.33e-08 ***
## sqrt(population)  9.160e-05  2.952e-05   3.103   0.0024 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.317 on 118 degrees of freedom
## Multiple R-squared:  0.8096, Adjusted R-squared:  0.8032
## F-statistic: 125.4 on 4 and 118 DF,  p-value: < 2.2e-16
```

Let's predict the test data.

```
pred_value_lm_sqrt = predict(sqrt_value_lm, newdata = value_test)
mean((pred_value_lm_sqrt - value_test$log_value)^2)
```

```
## [1] 0.1314996
```

This model produced a mean squared error of 0.13. Same as our previous model.
Let's try squaring the predictors besides year.

```
sqrd_value_lm = lm(log_value ~ - value + year + (revenue)^2 +
                     (yoy_val_inc)^2 + (debt_value)^2 +
                     (income)^2 + (population)^2, value_train)
summary(sqrd_value_lm)
```

```
##
## Call:
## lm(formula = log_value ~ -value + year + (revenue)^2 + (yoy_val_inc)^2 +
##     (debt_value)^2 + (income)^2 + (population)^2, data = value_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89167 -0.20541  0.00541  0.18494  0.94889
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.636e+02  7.665e+01   4.744 6.02e-06 ***
## year        -1.698e-01  3.792e-02  -4.479 1.77e-05 ***
## revenue      4.221e-09  2.920e-10  14.457  < 2e-16 ***
## yoy_val_inc  6.594e-01  2.955e-01   2.232  0.02757 *
## debt_value  -1.514e+00  2.454e-01  -6.169 1.03e-08 ***
## income      -1.320e-09  6.703e-10  -1.969  0.05131 .
## population   1.522e-08  5.753e-09   2.646  0.00927 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3284 on 116 degrees of freedom
## Multiple R-squared:  0.799,  Adjusted R-squared:  0.7887
## F-statistic: 76.87 on 6 and 116 DF,  p-value: < 2.2e-16
```

All predictors have p-values less than 0.05.
Predict test data.

```
pred_value_lm_sqrd = predict(sqrd_value_lm, newdata = value_test)
mean((pred_value_lm_sqrd - value_test$log_value)^2)
```

```
## [1] 0.1388793
```

The mean squared error on this model was also 0.13. No improvement on this transformation either.

Let's now try a ridge regression model.

```
set.seed(1)
train_matrix = model.matrix(log_value ~. - value, value_train)
test_matrix = model.matrix(log_value ~. - value, value_test)
# Now we need to select lambda using cross-validation
cv_out = cv.glmnet(train_matrix, value_train$log_value, alpha = 0)
best_lam = cv_out$lambda.min
best_lam
```

```
## [1] 0.05724196
```

```r
# Lambda chosen by cross-validation is 0.0572
# Now we fit ridge regression model and make predictions:
value_ridge = glmnet(train_matrix, value_train$log_value, alpha = 0)
pred_value_ridge = predict(value_ridge, s = best_lam, newx = test_matrix)
# Find test error:
mean((pred_value_ridge - value_test$log_value)^2)
```

```
## [1] 0.1338698
```

Our ridge regression model produced a mean squared error rate of 0.134. No improvement on this model either.
Let's try fitting a lasso regression model.

```r
set.seed(1)
train_matrix = model.matrix(log_value ~. - value, value_train)
test_matrix = model.matrix(log_value ~. - value, value_test)
# Now we need to select lambda using cross-validation
cv_out = cv.glmnet(train_matrix, value_train$log_value, alpha = 1)
best_lam = cv_out$lambda.min
best_lam
```

```
## [1] 0.001630272
```

```r
# Lambda chosen by cross-validation is 0.0014
# Now we fit ridge regression model and make predictions:
value_lasso = glmnet(train_matrix, value_train$log_value, alpha = 1)
pred_value_lasso = predict(value_lasso, s = best_lam, newx = test_matrix)
# Find test error:
mean((pred_value_lasso - value_test$log_value)^2)
```

```
## [1] 0.1392502
```

Our lasso regression model produced a mean squared error rate of approximately 0.14. Once again, no improvement.

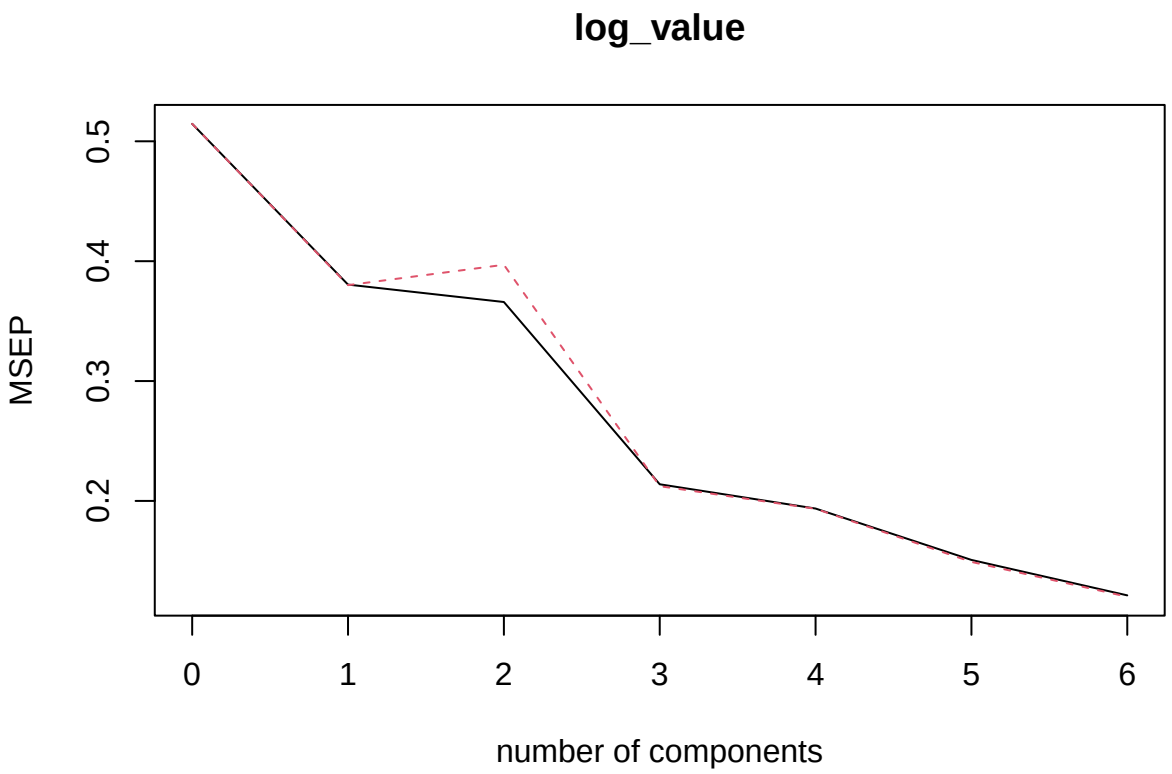
Let's try a principal component regression model.

```r
set.seed(1)
value_pcr = pcr(log_value ~. - value, data = value_train,
                scale = TRUE, validation = "CV")
summary(value_pcr)
```

```
## Data:    X dimension: 123 6
##  Y dimension: 123 1
## Fit method: svdpc
## Number of components considered: 6
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.7173   0.6168   0.6049   0.4625   0.4400   0.3883   0.3482
```

```
## adjCV       0.7173   0.6165   0.6301   0.4608   0.4398   0.3862   0.3465
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X            42.12    60.19    76.69    86.48    94.96    100.0
## log_value    27.11    28.54    62.55    67.18    75.47     79.9
```

```
par(mfrow = c(1, 1))
validationplot(value_pcr, val.type = "MSEP")
```



We see that where M = 6 and 5, our model produces the lowest mean squared error prediction value. However, we are trying our best to reduce the dimensions of the model.

```
pred_pcr = predict(value_pcr, value_test, ncomp = 4)
mean((pred_pcr - value_test$log_value)^2)
```

```
## [1] 0.1951959
```

When we fit a principal component regression using M = 4, we get a mean squared test error of 0.18.

```
# Now try using 3 components
pred_pcr = predict(value_pcr, value_test, ncomp = 3)
mean((pred_pcr - value_test$log_value)^2)
```

```
## [1] 0.1710488
```

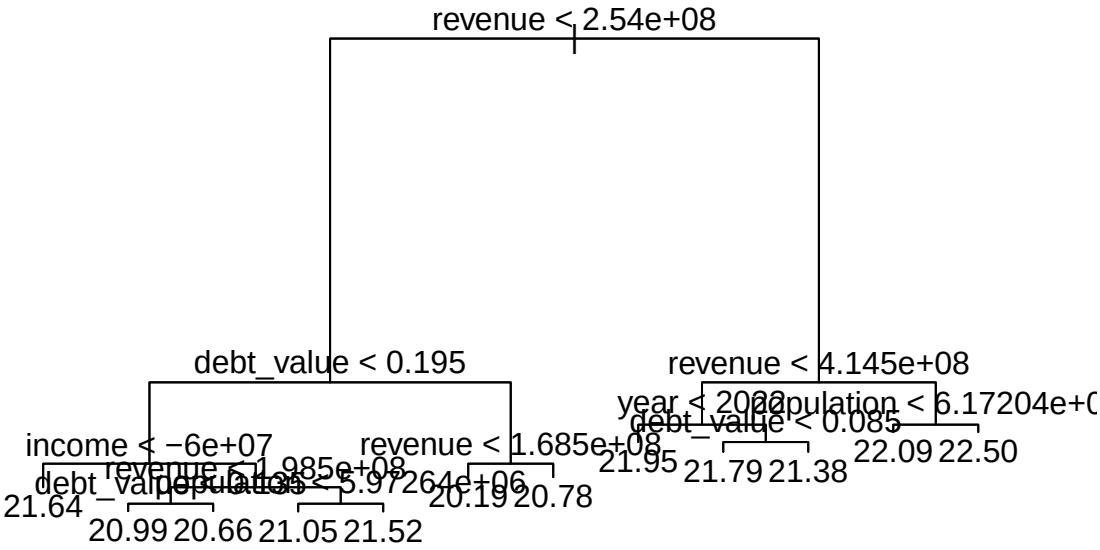
When we use M = 3, we get a mean squared error of 0.17. Both of our principal component mean squared error rates were no better than where we already are.

Let's try a regression tree.

```
value_tree = tree(log_value ~. - value, value_train)
summary(value_tree)
```

```
##
## Regression tree:
## tree(formula = log_value ~ . - value, data = value_train)
## Variables actually used in tree construction:
## [1] "revenue"    "debt_value" "income"     "population" "year"
## Number of terminal nodes:  12
## Residual mean deviance:  0.06562 = 7.284 / 111
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.68950 -0.14190 -0.02074  0.00000  0.12860  1.11100
```

```
plot(value_tree)
text(value_tree, pretty = 0)
```



There are 11 terminal nodes.
Let's predict our test data.

```
yhat = predict(value_tree, newdata = value_test)
mean((yhat - value_test[, "log_value"])^2)
```

```
## [1] 0.141275
```

This model produced a mean squared error of 0.14.

Let's try a random forest model.

```
value_rf = randomForest(log_value ~. - value, data = value_train, ntree = 500)
value_rf
```

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

```
##  randomForest(formula = log_value ~ . - value, data = value_train,      ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.08060061
##                     % Var explained: 84.08
```

Uskng 500 trees, this model explained 84.2% of our data. Let's predict our test data.

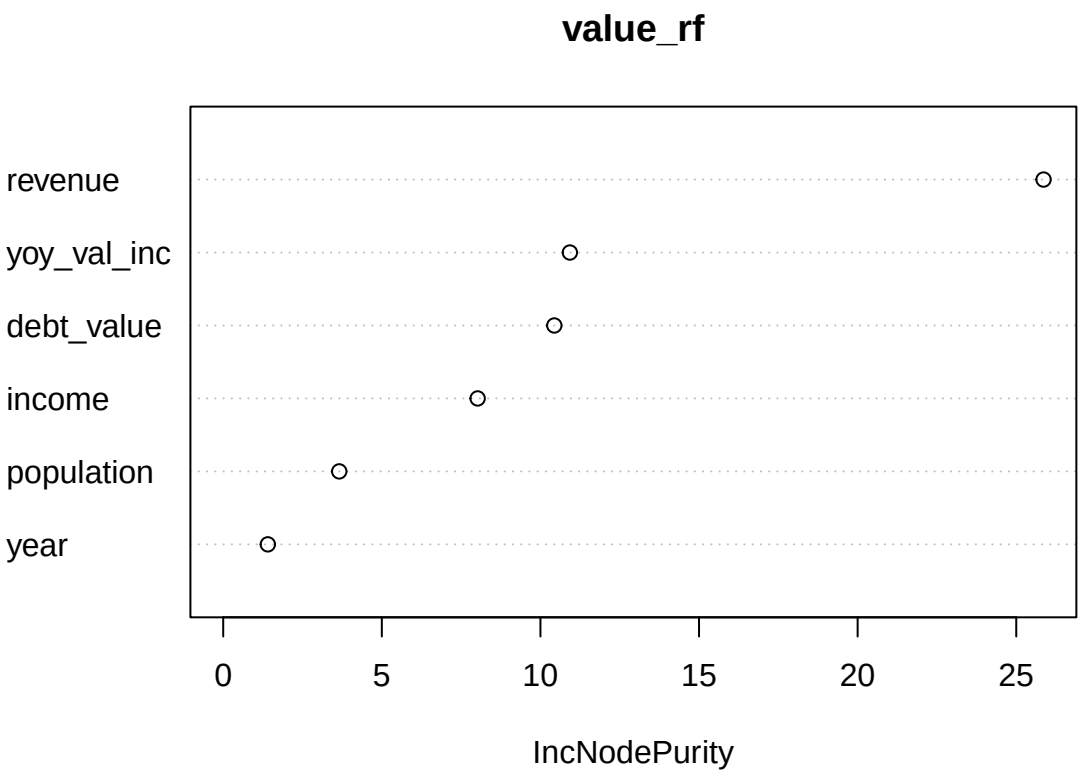
```
yhat_rf = predict(value_rf, newdata = value_test)
mean((yhat_rf - value_test$log_value)^2)
```

```
## [1] 0.06393874
```

Our random forest model produced a mean squared error of 0.064. This is by far our lowest mean squared error, meaning that the random forest model that we fit best represents our data when it comes to predicting the value response variable.

Let's look at most important variables in random forest model.

```
varImpPlot(value_rf)
```



The most important variables in random forest model are revenue and year over year value increase.

Conclusions Drawn from Analysis

Based on my exploratory analysis, my recommendation would be to at least invest in one NFL franchise. The league as a whole seems to generating the most revenue which is driving the values up year over year.

The NFL is also showing the lowest debt value of all sports in 2023. I think it is important to diversify the portfolio of our investors, so I would like to recommend investing in another franchise in another league as well.
Let's figure out what NFL franchise we will recommend to our investors. Let's filter our data to only have NFL franchises in 2023 since that is the most recent year.

```
nfl_value = value_df %>%
  filter(year == 2023 & league == "NFL")
# Since we only have 4 billion to spend, let's filter data to show teams only
# under 4 billion in value
nfl_value = nfl_value %>%
  filter(value < 4e9)
# All NFL values under $4 billion are over $3 billion
min(nfl_value$value)
```

```
## [1] 3e+09
```

The minimum value of an NFL franchise in 2023 is $3 billion.
Let's invest in the NFL team that is under $4 billion with the highest year over year value increase in 2023.

```
max_yoy_index = which(nfl_value$yoy_val_inc == max(nfl_value$yoy_val_inc))
nfl_value$team[max_yoy_index]
```

```
## [1] "Buffalo Bills"
```

```
nfl_value$value[nfl_value$team == "Buffalo Bills"]
```

```
## [1] 3.4e+09
```

This returns the Buffalo Bills who had a yoy value increase of 0.5 in 2023.
Assuming we purchase the Bills for their 2023 value, that would cost us $3.4 billion.
This leave us $600 million to invest in another franchise. Let's look at all of the other non-NFL franchises and see who had the largest year over year value increase.
We will want to make sure their values are less than or equal to $600 million.

```
non_nfl_value_df = value_df %>%
  filter(year == 2023 & league != "NFL")
non_nfl_value_df = non_nfl_value_df %>%
  filter(value < (4e9 - 3.4e9))
head(non_nfl_value_df)
```

```
##   year league             team  revenue   value yoy_val_inc debt_value  income
## 1 2023    NHL  Arizona Coyotes 1.27e+08 4.5e+08        0.12       0.69 5800000
## 2 2023    NHL Florida Panthers 1.37e+08 5.5e+08        0.22       0.30 4700000
##   population rev_to_val_ratio
## 1    4717225        0.2822222
## 2    6138858        0.2490909
```

There are only two teams that fit this criteria and they are both NHL teams: the Arizona Coyotes and the Florida Panthers.
As we saw earlier, next to the NFL, the NHL had the next highest average year over year value. Therefore,

I think that is a good financial move to invest in an NHL franchise, as the league seems to be growing in popularity and in value. Between the Panthers and the Coyotes, the Panthers have a higher year over year value increase and a lower debt value. Therefore, we can invest in the Panthers.

```
non_nfl_value_df$value[non_nfl_value_df$team == "Florida Panthers"]
```

```
## [1] 5.5e+08
```

Assuming we can purchase the Panthers for the value of $550 million, we would be able to do so with $50 million leftover from our initial $4 billion investment fund.

Conclusion Summary

We were able to use some regression tactics to fit models to our training data and predict the test data and check our mean squared error rates of those predictions. In doing so, we found that fitting a random forest model with 11 terminal nodes produced the lowest mean squared error rate of all of our models with revenue and the year-over-year value increase predictors being the most important ones of them all.
Using some exploratory analysis of the data set, we decided that the NFL is where all of the money is, based on value, year over year value increases, low debt values and high incomes. Therefore, I think it is smart to invest in a NFL franchise as it is the most profitable league. However, I think it is important to diversify our portfolio and invest in another team as well. After looking at the NFL teams that are valued at less than $4 billion, I suggested that they invest in the NFL franchise that had the highest year over year value increase, which happened to be the Buffalo Bills. With a new stadium coming in 2026 and a franchise quarterback located in a city that loves their football, I think that it is a sound investment to purchase the Buffalo Bills for $3.4 billion.
After deciding to invest in the Buffalo Bills, we were left with $600 million to invest in another franchise. The only other two franchises to invest in at that price point were the Florida Panthers and the Arizona Coyotes. Based on what we saw earlier in our analysis, we saw that the NHL had the second highest average year over year value increase next to the NFL. This shows that the NHL is growing in popularity which will drive revenue and increase the value of the league and individual franchises over time. Between the two franchises, the Panthers had a higher year over year value increase compared to the Coyotes. Plus, the Panthers have about half as less debt compared to the Coyotes. With a trip to the NHL Stanley Cup Finals last year as an eight seed going into the playoffs, I think this investment is a wise one and allows us to enter the NHL market.