STAT 425 Final Project: Understanding the most influential skills in soccer players relating to their market value

Yixuan Deng, Monte Thomas, Charan Govarthanaraj

2023-12-11

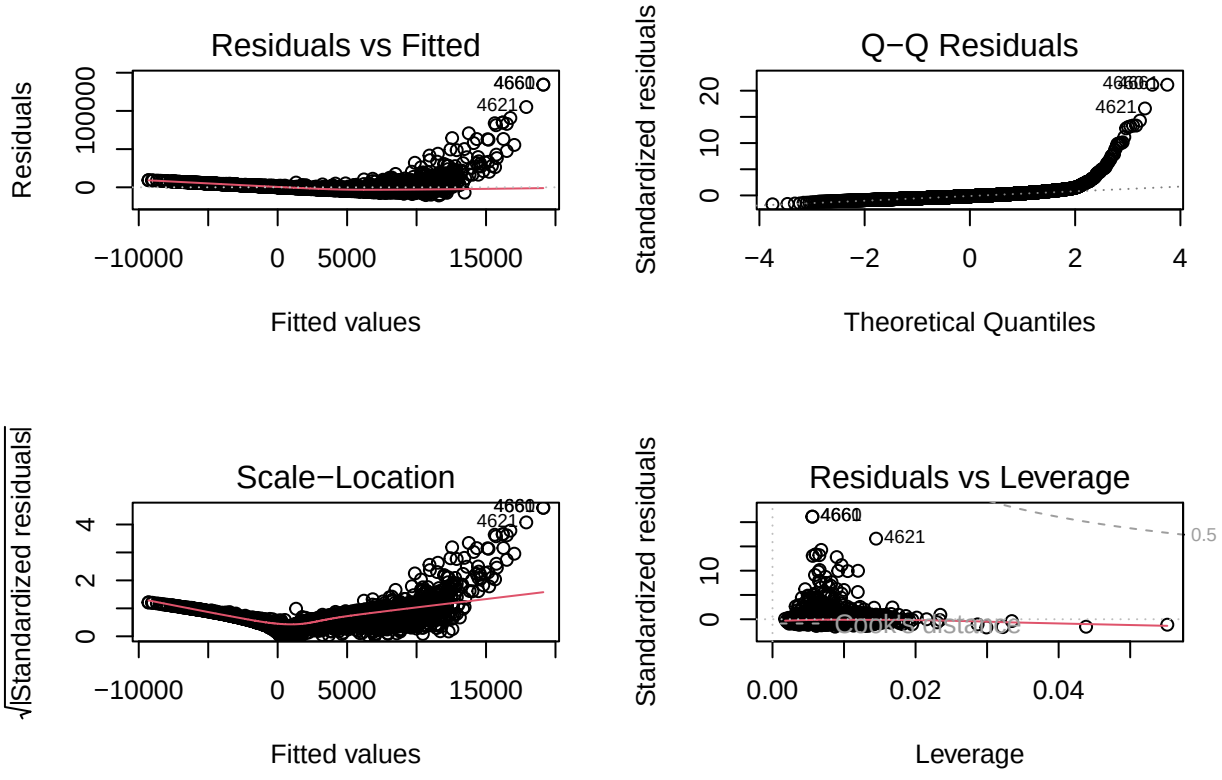
In this report, we delve into the extent to which a player's value is affected by each of his characteristics. We have selected a player's market value as the dependent variable, while the player's various physical metrics serve as the independent variables.

```
full_model = lm(value ~ . -player - country, data = project_data_1)
summary(full_model)
```

```
##
## Call:
## lm(formula = value ~ . - player - country, data = project_data_1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10625  -2652   -895   1212 134378
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -3.219e+04  4.492e+03  -7.166 8.73e-13 ***
## height          4.573e+01  2.481e+01   1.844  0.06528 .
## weight         -1.931e+01  2.089e+01  -0.924  0.35538
## age            -2.869e+02  2.354e+01 -12.188  < 2e-16 ***
## ball_control    1.203e+01  2.424e+01   0.496  0.61978
## dribbling      -1.034e+01  1.979e+01  -0.523  0.60116
## slide_tackle   -2.344e+01  2.147e+01  -1.092  0.27495
## stand_tackle    3.594e+01  2.227e+01   1.614  0.10664
## aggression      1.518e+01  1.025e+01   1.481  0.13873
## reactions       3.208e+02  1.754e+01  18.296  < 2e-16 ***
## att_position   -2.241e+01  1.517e+01  -1.477  0.13960
## interceptions  -2.173e+01  1.565e+01  -1.388  0.16520
## vision          2.923e+01  1.381e+01   2.117  0.03428 *
## composure       6.205e+01  1.495e+01   4.151 3.36e-05 ***
## crossing        3.751e+01  1.290e+01   2.908  0.00366 **
## short_pass      2.332e+01  2.418e+01   0.964  0.33493
## long_pass       3.361e+00  1.716e+01   0.196  0.84473
## acceleration    1.202e+01  1.781e+01   0.675  0.49968
## stamina         5.655e-01  1.057e+01   0.053  0.95734
## strength        8.571e+00  1.257e+01   0.682  0.49545
## balance         8.088e+00  1.284e+01   0.630  0.52864
## sprint_speed    1.899e+01  1.549e+01   1.226  0.22038
## agility        -1.711e+01  1.401e+01  -1.222  0.22195
## jumping         5.696e-01  9.149e+00   0.062  0.95036
```

```
## heading          2.847e+01  1.301e+01   2.188  0.02873 *
## shot_power      -4.812e+00  1.444e+01  -0.333  0.73891
## finishing        4.009e+01  1.625e+01   2.467  0.01367 *
## long_shots      -4.435e+01  1.550e+01  -2.862  0.00423 **
## curve            2.169e+01  1.374e+01   1.579  0.11444
## fk_acc           7.816e+00  1.243e+01   0.629  0.52937
## penalties       -1.118e+01  1.296e+01  -0.863  0.38825
## volleys          3.236e+01  1.334e+01   2.427  0.01528 *
## gk_positioning   2.551e+01  2.643e+01   0.965  0.33439
## gk_diving        4.252e+01  2.624e+01   1.620  0.10519
## gk_handling      3.723e+01  2.663e+01   1.398  0.16211
## gk_kicking       6.190e+00  2.497e+01   0.248  0.80420
## gk_reflexes      1.694e+01  2.602e+01   0.651  0.51506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6377 on 5645 degrees of freedom
## Multiple R-squared:  0.2914, Adjusted R-squared:  0.2869
## F-statistic:  64.5 on 36 and 5645 DF,  p-value: < 2.2e-16
```

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




```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```r
boxcox_model <- boxcox(full_model, plotit = TRUE)
lambda <- boxcox_model$x[which.max(boxcox_model$y)]
print(lambda)
```

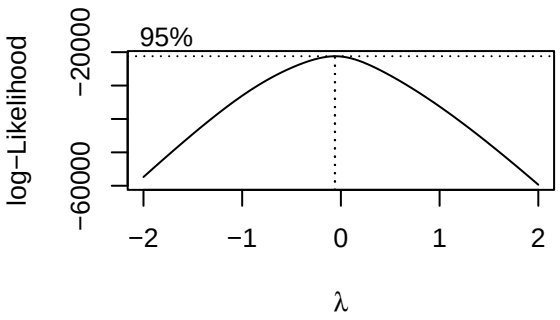
```
## [1] -0.06060606
```

```r
transformed_model <- lm(log(value) ~ . -player - country, data = project_data_1)
summary(transformed_model)
```

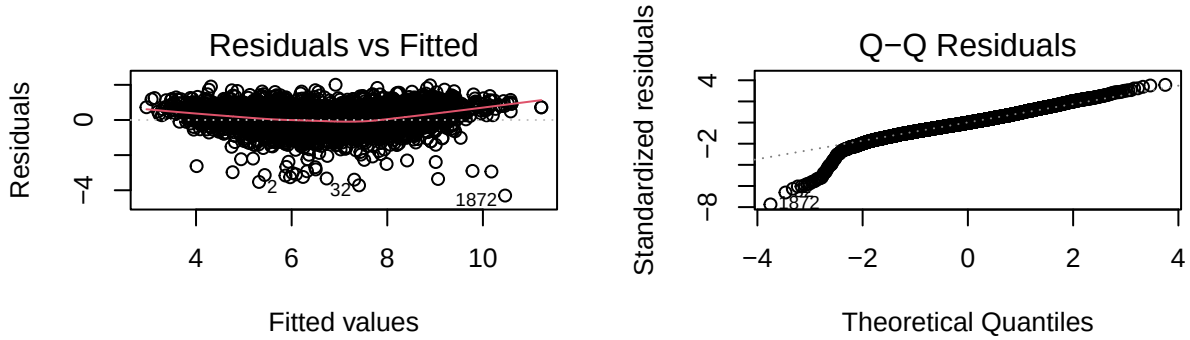
```
##
## Call:
## lm(formula = log(value) ~ . - player - country, data = project_data_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3016 -0.3215 -0.0086  0.3385  2.0039
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -3.1728279  0.3977152  -7.978 1.79e-15 ***
## height         -0.0002320  0.0021965  -0.106 0.915871
## weight          0.0020924  0.0018497   1.131 0.257999
## age            -0.0546164  0.0020841 -26.206  < 2e-16 ***
## ball_control    0.0272051  0.0021463  12.675  < 2e-16 ***
## dribbling       0.0044239  0.0017521   2.525 0.011599 *
## slide_tackle   -0.0009949  0.0019008  -0.523 0.600711
## stand_tackle    0.0066890  0.0019722   3.392 0.000700 ***
## aggression     -0.0006787  0.0009080  -0.747 0.454800
## reactions       0.0691673  0.0015527  44.547  < 2e-16 ***
## att_position   -0.0081474  0.0013432  -6.066 1.40e-09 ***
## interceptions  -0.0037843  0.0013860  -2.730 0.006346 **
## vision         -0.0038868  0.0012224  -3.180 0.001482 **
## composure       0.0109027  0.0013237   8.237  < 2e-16 ***
## crossing        0.0105873  0.0011423   9.269  < 2e-16 ***
## short_pass      0.0242885  0.0021409  11.345  < 2e-16 ***
## long_pass      -0.0069861  0.0015194  -4.598 4.36e-06 ***
## acceleration    0.0056022  0.0015767   3.553 0.000384 ***
## stamina         0.0086235  0.0009361   9.212  < 2e-16 ***
## strength        0.0058309  0.0011132   5.238 1.68e-07 ***
## balance        -0.0026195  0.0011366  -2.305 0.021218 *
## sprint_speed    0.0058471  0.0013716   4.263 2.05e-05 ***
## agility        -0.0016468  0.0012405  -1.328 0.184386
## jumping        -0.0009034  0.0008101  -1.115 0.264812
## heading         0.0139069  0.0011522  12.069  < 2e-16 ***
## shot_power      0.0113660  0.0012785   8.890  < 2e-16 ***
## finishing       0.0057441  0.0014393   3.991 6.66e-05 ***
## long_shots     -0.0064712  0.0013721  -4.716 2.46e-06 ***
## curve           0.0008406  0.0012166   0.691 0.489643
## fk_acc         -0.0001933  0.0011003  -0.176 0.860580
## penalties      -0.0048482  0.0011475  -4.225 2.43e-05 ***
## volleys        -0.0003824  0.0011809  -0.324 0.746109
## gk_positioning  0.0141365  0.0023401   6.041 1.63e-09 ***
## gk_diving       0.0136936  0.0023236   5.893 4.00e-09 ***
## gk_handling     0.0098213  0.0023580   4.165 3.16e-05 ***
```

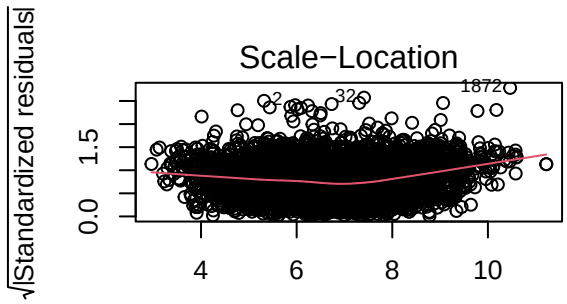
```
## gk_kicking       0.0026071  0.0022108   1.179 0.238352
## gk_reflexes      0.0157697  0.0023044   6.843 8.55e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5646 on 5645 degrees of freedom
## Multiple R-squared:  0.823,  Adjusted R-squared:  0.8219
## F-statistic: 729.1 on 36 and 5645 DF,  p-value: < 2.2e-16
```

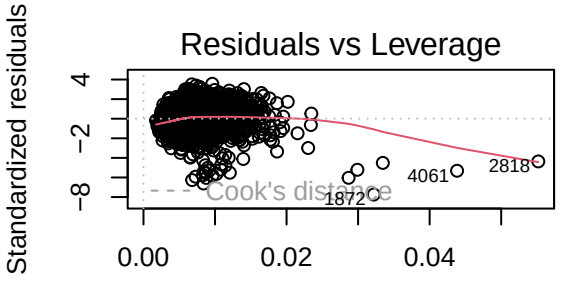
```
par(mfrow = c(2,2))
```




```
plot(transformed_model)
```








```
p_values <- summary(transformed_model)$coefficients[, "Pr(>|t|)"]
print(p_values)
```

```
##    (Intercept)         height         weight            age   ball_control
##   1.792223e-15   9.158709e-01   2.579989e-01  5.957939e-143   2.520290e-36
##      dribbling   slide_tackle   stand_tackle     aggression      reactions
##   1.159912e-02   6.007105e-01   6.995672e-04   4.548001e-01   0.000000e+00
```

```
##   att_position  interceptions         vision      composure       crossing
##   1.397966e-09   6.345847e-03   1.482406e-03   2.180717e-16   2.622945e-20
##     short_pass      long_pass   acceleration        stamina       strength
##   1.634182e-29   4.359356e-06   3.837513e-04   4.414449e-20   1.684038e-07
##        balance   sprint_speed        agility        jumping        heading
##   2.121810e-02   2.048657e-05   1.843855e-01   2.648116e-01   3.911080e-33
##     shot_power      finishing     long_shots          curve         fk_acc
##   8.083781e-19   6.663312e-05   2.461336e-06   4.896427e-01   8.605796e-01
##      penalties        volleys gk_positioning      gk_diving    gk_handling
##   2.426625e-05   7.461088e-01   1.630328e-09   4.003665e-09   3.159463e-05
##     gk_kicking    gk_reflexes
##   2.383521e-01   8.554710e-12
```

```
significant_variables <- names(p_values[p_values < 0.05 & names(p_values) != "(Intercept)"])

print(significant_variables)
```

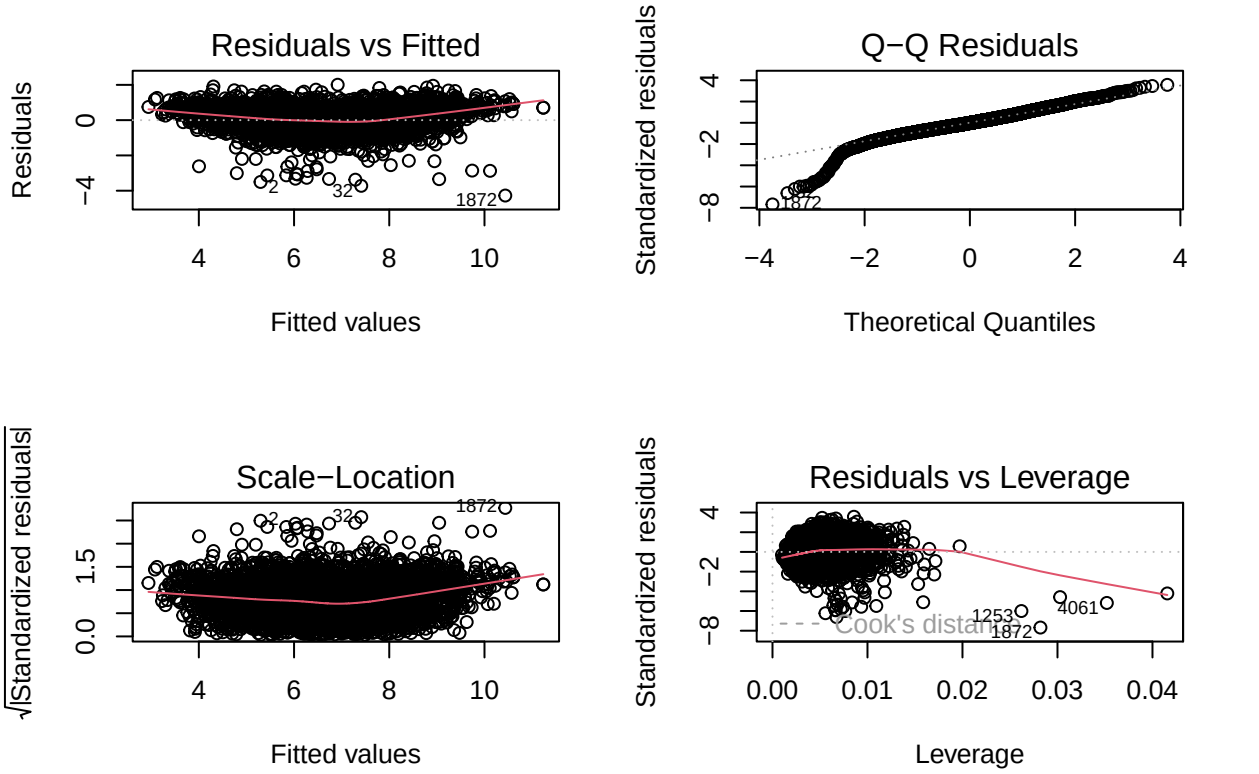
```
##  [1] "age"            "ball_control"   "dribbling"      "stand_tackle"
##  [5] "reactions"      "att_position"   "interceptions"  "vision"
##  [9] "composure"      "crossing"       "short_pass"     "long_pass"
## [13] "acceleration"   "stamina"        "strength"       "balance"
## [17] "sprint_speed"   "heading"        "shot_power"     "finishing"
## [21] "long_shots"     "penalties"      "gk_positioning" "gk_diving"
## [25] "gk_handling"    "gk_reflexes"
```

```
smallest_model <- lm(log(value) ~ .,
                     data = project_data_1[, c("value", significant_variables)])
summary(smallest_model)
```

```
##
## Call:
## lm(formula = log(value) ~ ., data = project_data_1[, c("value",
##     significant_variables)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2738 -0.3255 -0.0090  0.3386  2.0060
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -3.0784962  0.0992783 -31.009  < 2e-16 ***
## age            -0.0551523  0.0020381 -27.060  < 2e-16 ***
## ball_control    0.0272484  0.0021407  12.728  < 2e-16 ***
## dribbling       0.0042431  0.0017380   2.441 0.014661 *
## stand_tackle    0.0059277  0.0013436   4.412 1.04e-05 ***
## reactions       0.0691584  0.0015472  44.698  < 2e-16 ***
## att_position   -0.0081848  0.0013302  -6.153 8.12e-10 ***
## interceptions  -0.0041885  0.0013459  -3.112 0.001867 **
## vision         -0.0038985  0.0012124  -3.216 0.001309 **
## composure       0.0108349  0.0013050   8.303  < 2e-16 ***
## crossing        0.0109053  0.0010135  10.761  < 2e-16 ***
## short_pass      0.0243657  0.0021372  11.401  < 2e-16 ***
## long_pass      -0.0070387  0.0015058  -4.674 3.02e-06 ***
## acceleration    0.0048928  0.0015131   3.234 0.001229 **
## stamina         0.0082553  0.0009222   8.951  < 2e-16 ***
```

```
## strength         0.0062065  0.0009562   6.491 9.26e-11 ***
## balance         -0.0036555  0.0009414  -3.883 0.000104 ***
## sprint_speed     0.0056933  0.0013639   4.174 3.03e-05 ***
## heading          0.0133692  0.0010497  12.736  < 2e-16 ***
## shot_power       0.0114445  0.0012434   9.204  < 2e-16 ***
## finishing        0.0057210  0.0013933   4.106 4.08e-05 ***
## long_shots      -0.0064949  0.0012766  -5.087 3.75e-07 ***
## penalties       -0.0047133  0.0010797  -4.365 1.29e-05 ***
## gk_positioning   0.0146383  0.0022847   6.407 1.60e-10 ***
## gk_diving        0.0141893  0.0022667   6.260 4.13e-10 ***
## gk_handling      0.0105906  0.0022968   4.611 4.09e-06 ***
## gk_reflexes      0.0163196  0.0022534   7.242 5.01e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5645 on 5655 degrees of freedom
## Multiple R-squared:  0.8228, Adjusted R-squared:  0.8219
## F-statistic:  1010 on 26 and 5655 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2,2))
plot(smallest_model)
```



Based on the previous analysis, our team has elaborated on the advantages of using the natural logarithm of a player's value as a variable. Therefore, in all subsequent regression analyses, we will use the natural logarithm of player's value for the study in order to obtain more accurate and insightful analysis results.

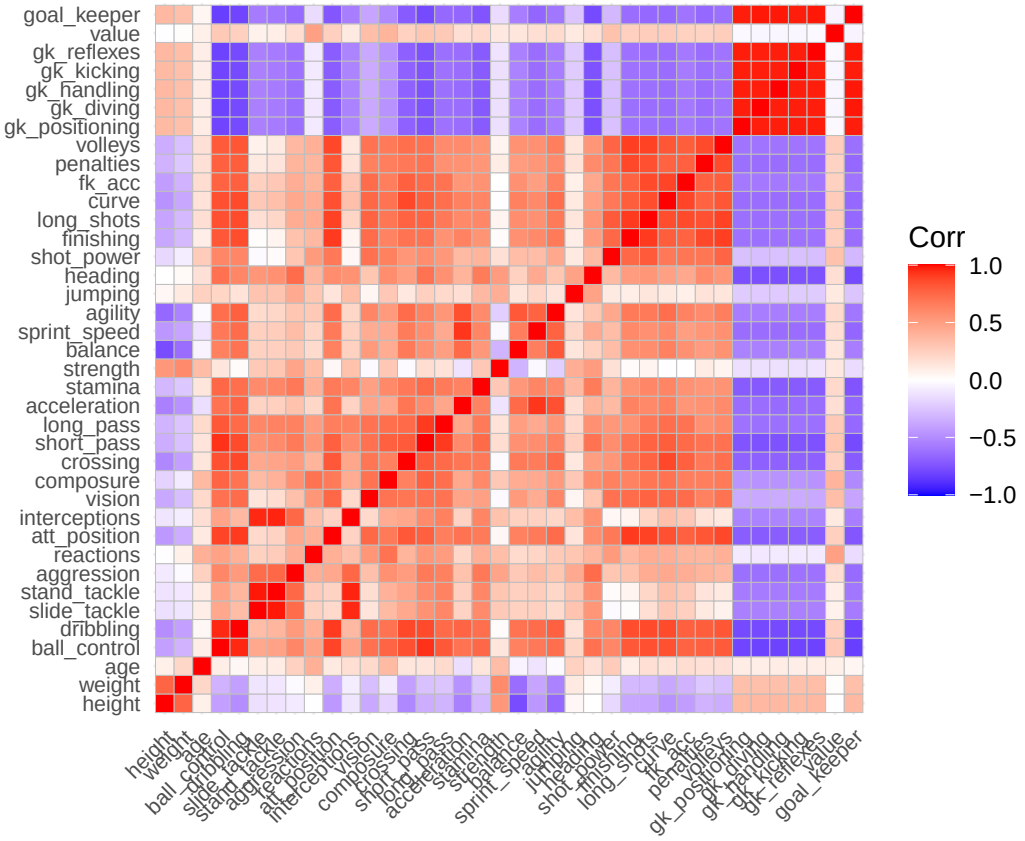
```r
project_data_1$country <- as.factor(project_data_1$country)
project_data_1$goal_keeper <- 0
project_data_1[project_data_1$gk_reflexes>40,]$goal_keeper <- 1
y <- log(project_data_1$value)
```

```
x <- project_data_1[,-which(names(project_data_1)=="value" | names(project_data_1)=="player")]
```

Before conducting regression analysis, our first step is to analyze the distribution pattern of the independent variable (X) in depth. Here, we examine the correlation coefficients between the independent variables by constructing a correlation matrix. The main idea is to identify potential multicollinearity problems, which is a key prerequisite check in regression analysis.

The presence of multicollinearity may make ordinary least squares (OLS) estimation problematic or even unsolvable. Even if it does not result in complete multicollinearity, if there is a high degree of correlation between the variables, this can lead to a significant increase in the variance of the estimate. This expansion of variance can weaken the accuracy of statistical hypothesis testing and affect the reliability of model estimation. Therefore, identifying and dealing with these correlations is critical to ensure the validity and accuracy of regression analysis.

```
numerical_data <- select_if(project_data_1, is.numeric)
cor_matrix <- cor(numerical_data)
ggcorrplot(cor_matrix, method ="square", lab_size = TRUE)+theme(axis.text.x = element_text(size=8),
                                                                axis.text.y = element_text(size=8))
```

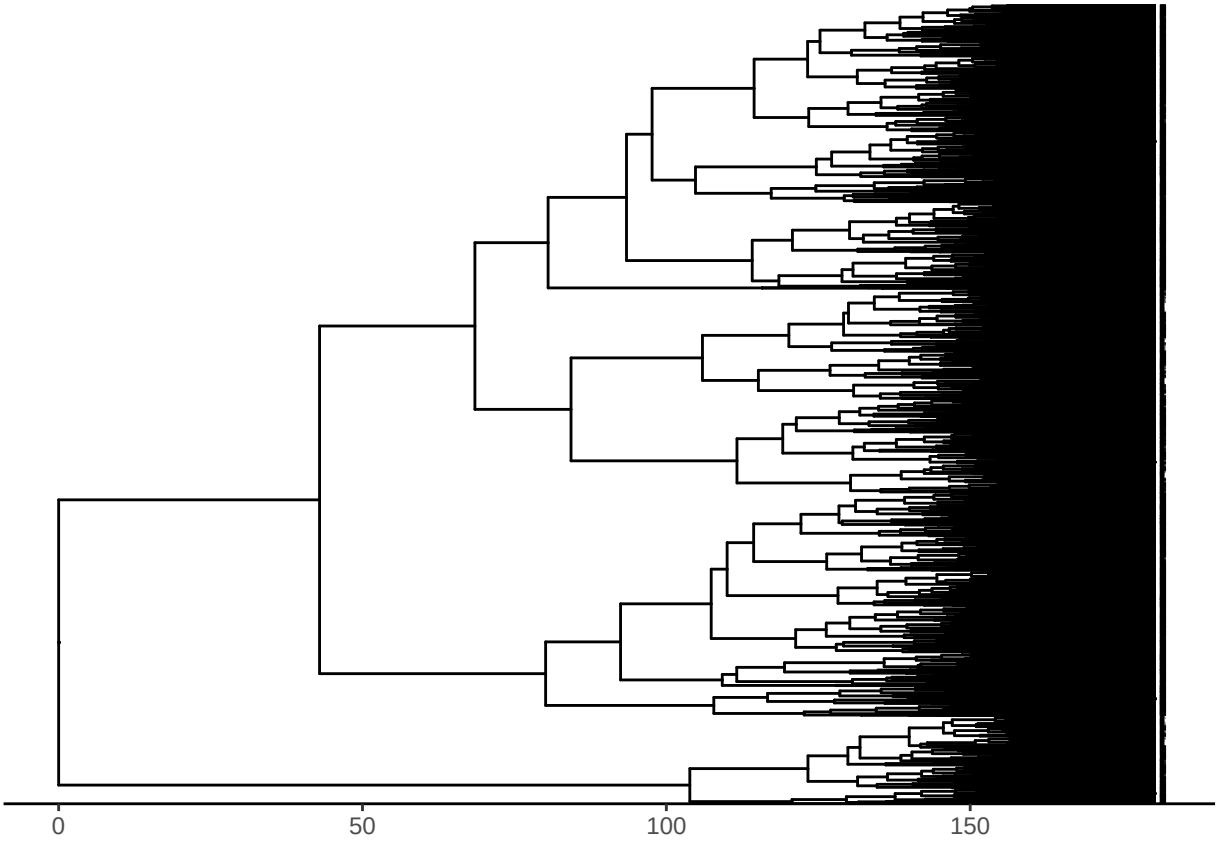


From this figure we can clearly see that the dark red areas and the dark purple areas represent issues where there is a high correlation between some of the variables, so further variable selection, either manually or using automated selection (e.g., lasso as well as the ridge method) is necessary.

Next, we consider the use of clustering analysis to further analyze whether clustering exists between X's.

```
x[,-1] <- lapply(x[,-1], as.numeric)
numerical_x <- select_if(x, is.numeric)
#Perform clustering
dist_matrix <- dist(numerical_x)
```

```
hc <- hclust(dist_matrix)
tree <- as.phylo(hc)
ggtree(tree) +
  geom_tiplab(size = 0.5) +
  theme_tree2()
```

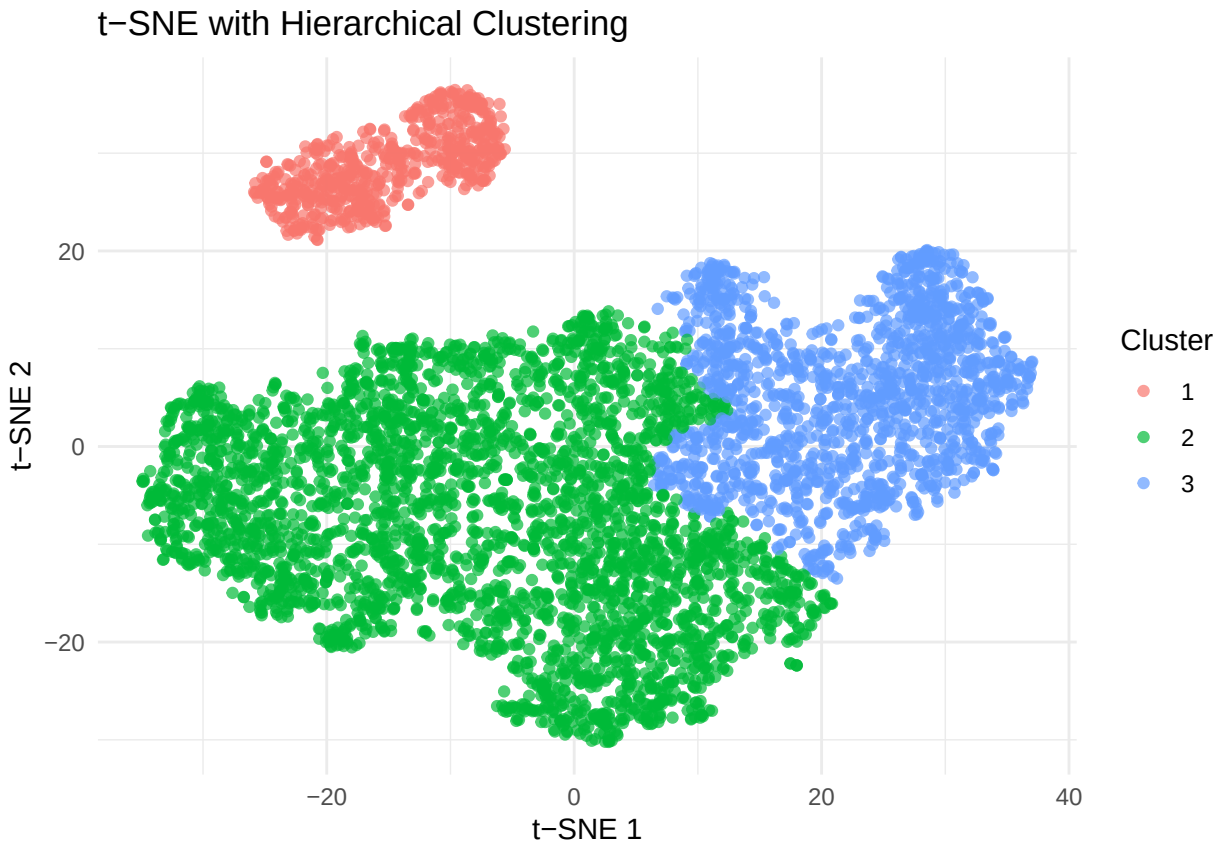


We first use hierarchical clustering for our analysis, as one can see from this figure above, it is difficult to accurately represent the results of hierarchical clustering using a tree diagram because of the excessive number of variables.

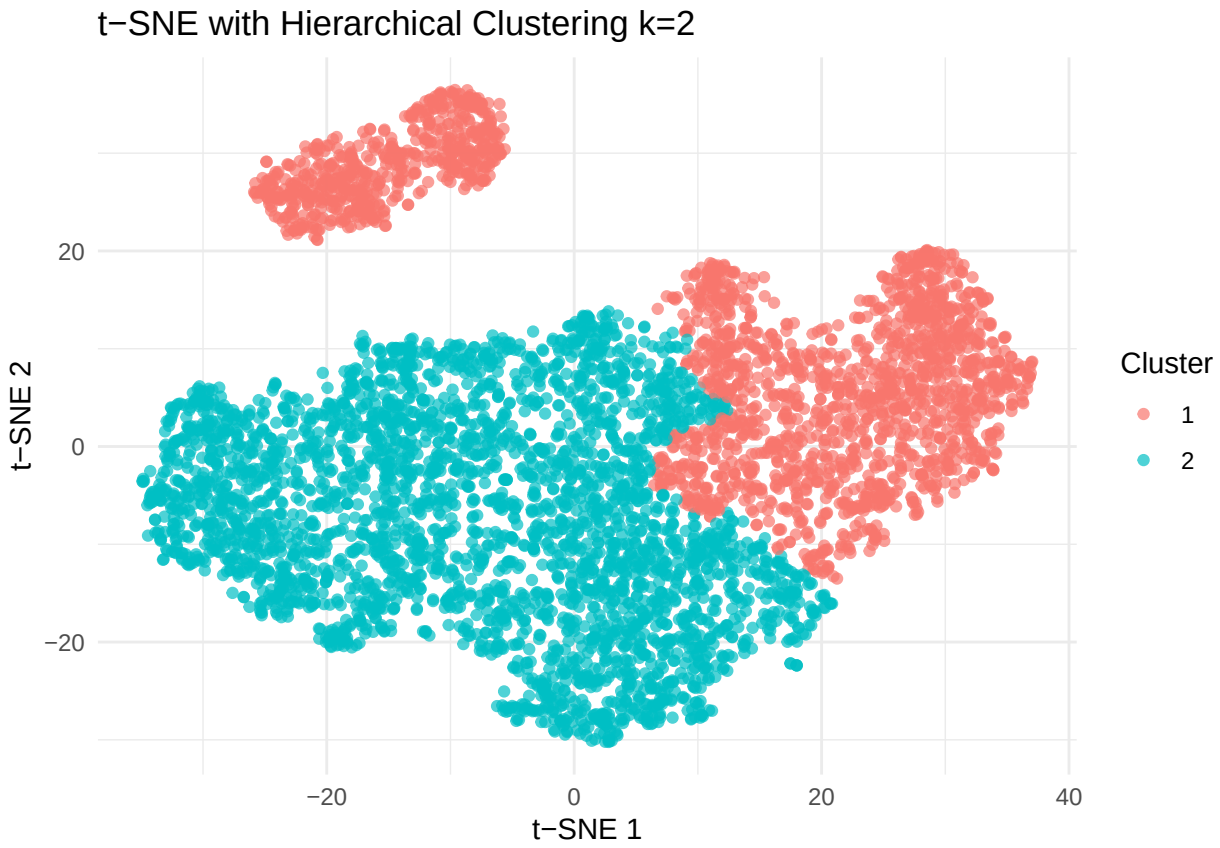
Therefore, we consider using the tsne plot to compress the high-dimensional data into two dimensions to better visualize the results of this cluster.

```
library("Rtsne")
tsne_results <- Rtsne(unique(numerical_x),
                      dims = 2, perplexity = 30)
hc <- hclust(dist(tsne_results$Y))
```

```
clusters <- cutree(hc, k = 3)
tsne_df <- as.data.frame(tsne_results$Y)
tsne_df$cluster <- as.factor(clusters)
ggplot(tsne_df, aes(x = V1, y = V2, color = cluster)) +
  geom_point(alpha = 0.7) +
  theme_minimal() +
  labs(title = "t-SNE with Hierarchical Clustering",
       x = "t-SNE 1",
       y = "t-SNE 2",
       color = "Cluster")
```

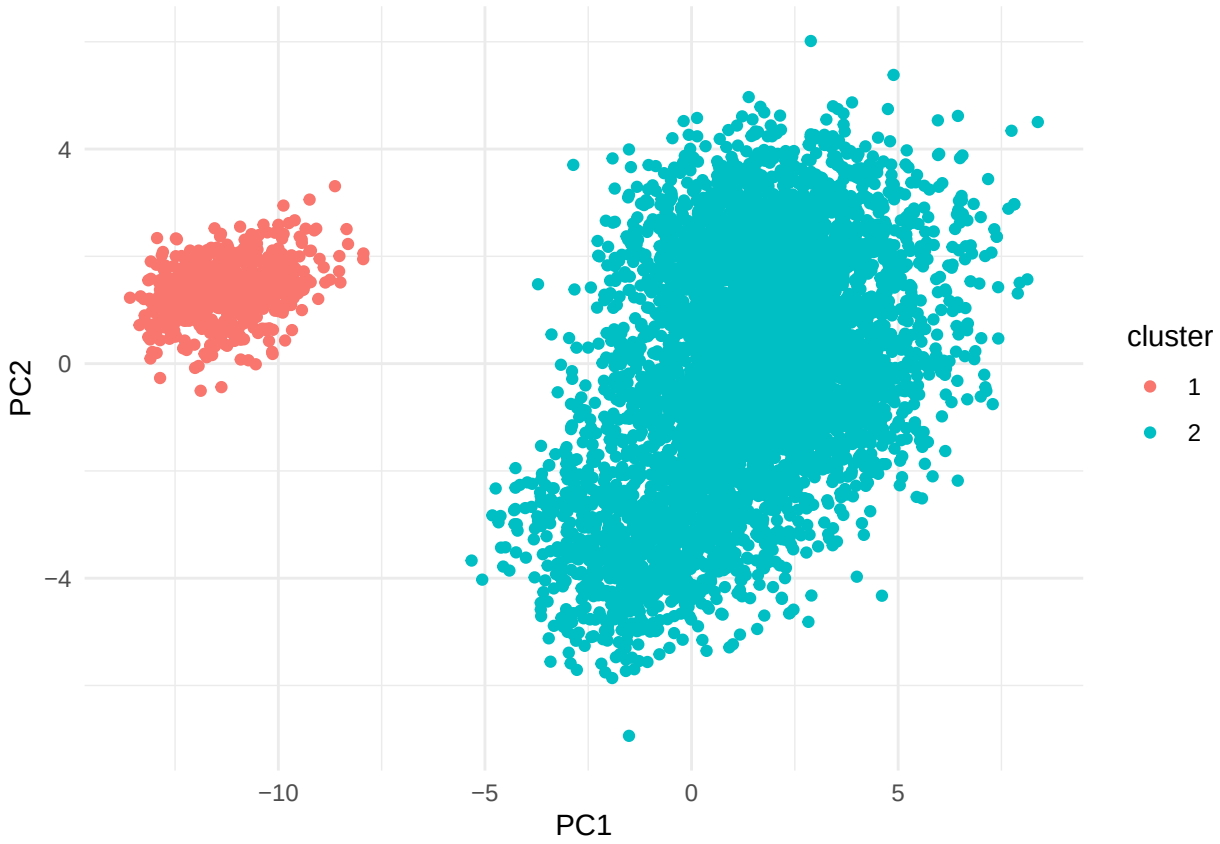



```r
clusters <- cutree(hc, k = 2)
tsne_df <- as.data.frame(tsne_results$Y)
tsne_df$cluster <- as.factor(clusters)
ggplot(tsne_df, aes(x = V1, y = V2, color = cluster)) +
  geom_point(alpha = 0.7) +
  theme_minimal() +
  labs(title = "t-SNE with Hierarchical Clustering k=2",
       x = "t-SNE 1",
       y = "t-SNE 2",
       color = "Cluster")
```



The above figure shows the result of using hierarchical cluster K=2 and K=3 in tnse respectively, we can see that in this result, using hierarchical cluster is not good enough to make a complete cluster of the data. therefore, we consider using kmeans in the next step. in the case of using kmeans, we first perform a PCA transformation on the data. The main purpose of PCA transformation is the same as that of tsne, which is to visualize the data in low dimensions.

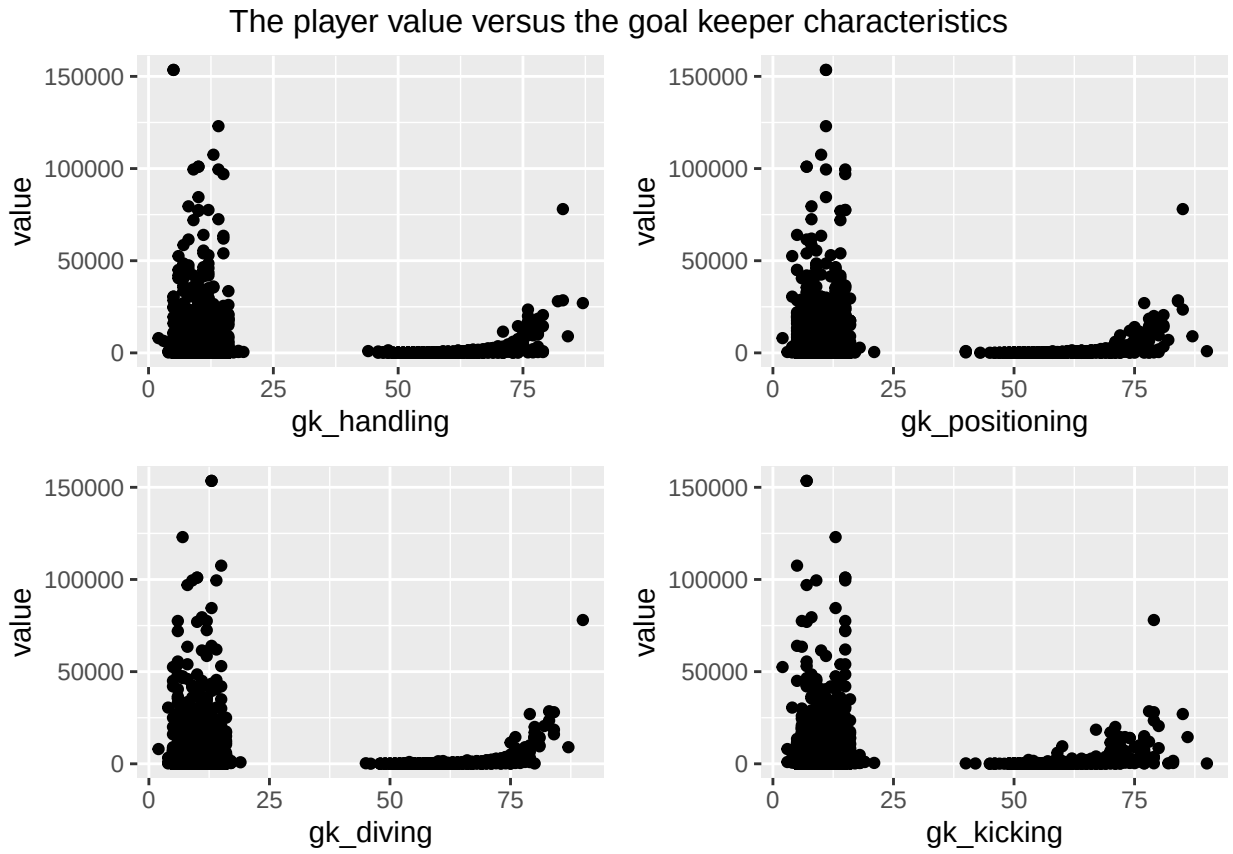
```
#could also use PCA to show the results.
pca_result <- prcomp(numerical_x, scale. = TRUE)
data_pca <- pca_result$x[, 1:2]
kmeans_result_pca <- kmeans(data_pca, centers = 2)
data_pca_df <- as.data.frame(data_pca)
data_pca_df$cluster <- factor(kmeans_result_pca$cluster)
ggplot(data_pca_df, aes(PC1, PC2, color = cluster)) +
  geom_point() +
  theme_minimal()
```



Another approach is to use PCA for dimensionality reduction and then select the first two principal components for Kmeans analysis. The result on the figure shows the result of PCA analysis and the two clusters generated by Kmeans.

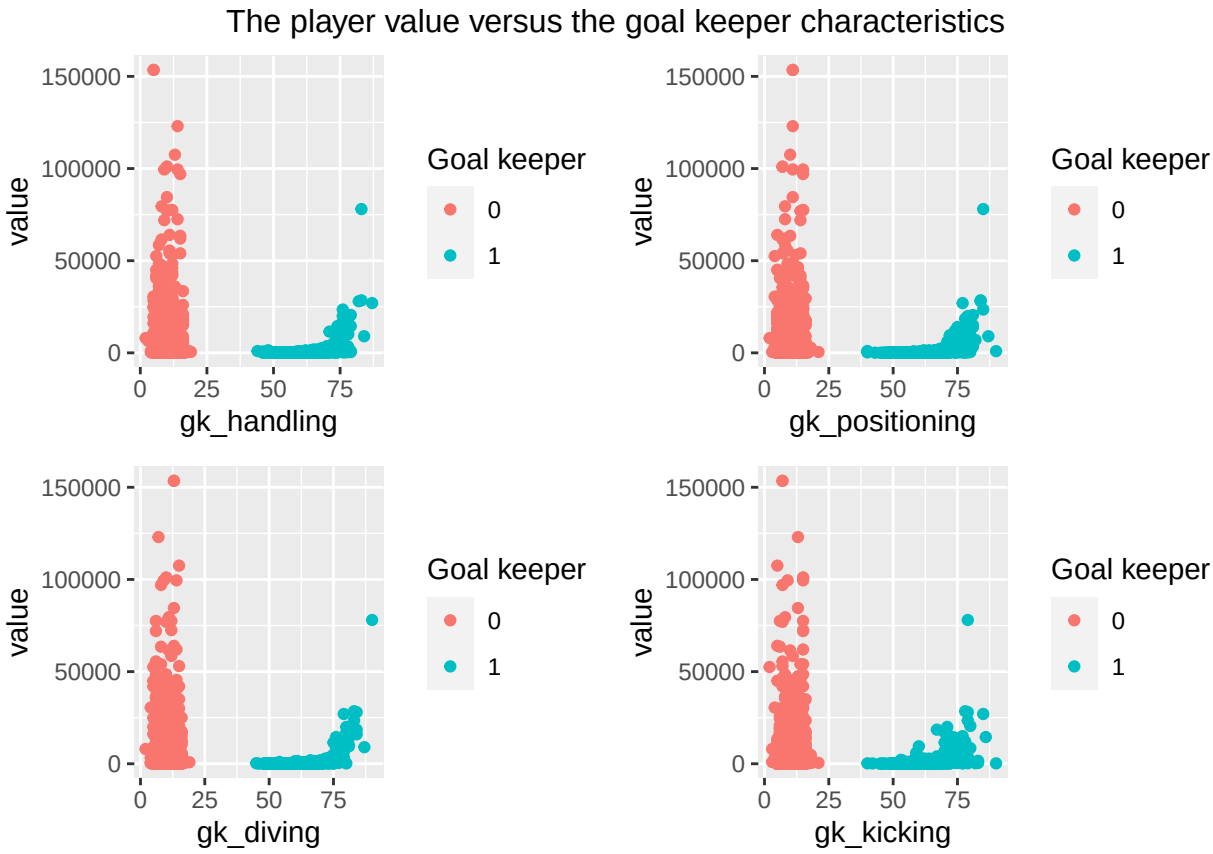
From both the t-SNE and the PCA with clustering analysis, we can see that there are indeed two main clusters that are not captured by any categorical variables. Generally, there should have a categorical variable that indicates the position of the player, however, in this dataset we didn't find it. Therefore, a reasonable assumption could be because of the goal keeper position that results in the two different clusters. Therefore, now we further check by plotting out the distribution of the goal keepers characteristics.

```
plota <- ggplot(project_data_1, aes(x=gk_handling, y=value))+geom_point()
plotb <- ggplot(project_data_1, aes(x=gk_positioning, y=value))+geom_point()
plotc <- ggplot(project_data_1, aes(x=gk_diving, y=value))+geom_point()
plotd <- ggplot(project_data_1, aes(x=gk_kicking, y=value))+geom_point()
grid.arrange(plota,plotb,plotc,plotd,top="The player value versus the goal keeper characteristics")
```



Therefore, we added a new dummy variable which is called goal keeper, and the value is 1 if the gk_reflex variable is larger than 40 and 0 if the value is smaller than 40.

```
plota <- ggplot(project_data_1, aes(x=gk_handling, y=value,
                                    colour=as.factor(goal_keeper)))+geom_point()+labs(colour="Goal keeper")
plotb <- ggplot(project_data_1, aes(x=gk_positioning, y=value,
                                    colour=as.factor(goal_keeper)))+geom_point()+labs(colour="Goal keeper")
plotc <- ggplot(project_data_1, aes(x=gk_diving, y=value,
                                    colour=as.factor(goal_keeper)))+geom_point()+labs(colour="Goal keeper")
plotd <- ggplot(project_data_1, aes(x=gk_kicking, y=value,
                                    colour=as.factor(goal_keeper)))+geom_point()+labs(colour="Goal keeper")
grid.arrange(plota,plotb,plotc,plotd,top="The player value versus the goal keeper characteristics")
```



Therefore, by using the clustering analysis, we successively recgonized that there is some potential categorical structure in the data and added one categorical variable.

Having learned that there is some degree of correlation and clustering in the data, this further justifies our use of the lasso and ridge methods. Next, we consider the analysis using lasso and ridge.

Here, we first randomly split the data into a 75% training dataset and a 25% test dataset. After that, we run four models on the training dataset, which are Lasso and Ridge regression with the best parameter obtained by 10-fold cross validation, a regression with manual variable selection, and a regression with all numerical variables, and a linear model with all variables.

After that, we train on each of these four models and then test them on the test set. And the RMSE is calculated to see the final predictive ability of the models by comparing the RMSE.

```
#First split the data into training and testing
set.seed(123)
index <- sample(1:nrow(numerical_x), size = nrow(numerical_x)*0.75)
training_x <- numerical_x[index,]
training_y <- y[index]
testing_x <- numerical_x[-index,]
testing_y <- y[-index]
```

Run Lasso and ridge regression.

```
cv_model_lasso <- cv.glmnet(as.matrix(training_x), training_y, alpha=1,
                            nfolds = 10)
cv_model_ridge <- cv.glmnet(as.matrix(training_x), training_y, alpha=0,
                            nfolds = 10)
```

```r
best_lambda_lasso <- cv_model_lasso$lambda.min
best_lambda_lasso
```

```
## [1] 0.0005904013
```

```r
final_model_lasso <- glmnet(as.matrix(training_x), training_y, alpha = 1,
                            lambda = best_lambda_lasso)

best_lambda_ridge <- cv_model_ridge$lambda.min
best_lambda_ridge
```

```
## [1] 0.1106304
```

```r
final_model_ridge <- glmnet(as.matrix(training_x), training_y, alpha = 0,
                            lambda = best_lambda_ridge)
coef(final_model_ridge)
```

```
## 38 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)    -3.3797700490
## height          0.0032548565
## weight          0.0021674075
## age            -0.0458056404
## ball_control    0.0154733172
## dribbling       0.0057191822
## slide_tackle    0.0013815333
## stand_tackle    0.0024863585
## aggression      0.0001965586
## reactions       0.0638777649
## att_position   -0.0023228604
## interceptions  -0.0006466526
## vision         -0.0008657096
## composure       0.0142852999
## crossing        0.0068425239
## short_pass      0.0159592790
## long_pass       0.0004080078
## acceleration    0.0066998906
## stamina         0.0069911223
## strength        0.0061223316
## balance        -0.0017788055
## sprint_speed    0.0066822635
## agility        -0.0008294244
## jumping        -0.0002321177
## heading         0.0104275995
## shot_power      0.0092471638
## finishing       0.0021570424
## long_shots     -0.0028228999
## curve           0.0009132096
## fk_acc         -0.0021017434
## penalties      -0.0017184079
## volleys         0.0002049623
## gk_positioning  0.0091261856
## gk_diving       0.0094148975
## gk_handling     0.0073865219
## gk_kicking      0.0062893010
```

```
## gk_reflexes     0.0097882277
## goal_keeper     0.4221382288
```

```
coef(final_model_lasso)
```

```
## 38 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)    -3.1892424345
## height         -0.0002011022
## weight          0.0014759019
## age            -0.0527517874
## ball_control    0.0276260952
## dribbling       0.0043588788
## slide_tackle    .
## stand_tackle    0.0056687317
## aggression     -0.0009004972
## reactions       0.0690601047
## att_position   -0.0071801325
## interceptions  -0.0028398773
## vision         -0.0036453515
## composure       0.0117943547
## crossing        0.0097547947
## short_pass      0.0216447501
## long_pass      -0.0058464291
## acceleration    0.0077156658
## stamina         0.0088677802
## strength        0.0059522256
## balance        -0.0022881709
## sprint_speed    0.0059780700
## agility        -0.0020340632
## jumping        -0.0015228984
## heading         0.0144274328
## shot_power      0.0102528275
## finishing       0.0048692288
## long_shots     -0.0052688759
## curve           .
## fk_acc         -0.0005606478
## penalties      -0.0033895984
## volleys        -0.0004936142
## gk_positioning  0.0119682149
## gk_diving       0.0114716512
## gk_handling     0.0047500341
## gk_kicking      .
## gk_reflexes     0.0144639857
## goal_keeper     0.8474137035
```

Now conduct variable selection manually

```
transformed_model <- lm(log(value) ~ . -player - country, data = project_data_1)

p_values <- summary(transformed_model)$coefficients[, "Pr(>|t|)"]
significant_variables <- names(p_values[p_values < 0.05 & names(p_values) != "(Intercept)"])



training_df <- as.data.frame(training_x)
```

```
training_df$y <- training_y
# fit a linear model for comparison reason
smallest_model2 <- lm(y ~ .,
                      data = training_df[, c("y", significant_variables)])
```

```
simple_linear_model <- lm(y~., data=training_df)
#summary(simple_linear_model)
#obtain linear, lasso and ridge calculation
lm_prediction <- predict(simple_linear_model,newdata = testing_x)
lm_best_select_prediction <- predict(smallest_model2,newdata = testing_x)

predictions_ridge <- predict(final_model_ridge, newx = as.matrix(testing_x))
predictions_lasso <- predict(final_model_lasso, newx = as.matrix(testing_x))
rmse_ridge <- sqrt(mean((testing_y - predictions_ridge)^2))
rmse_lasso <- sqrt(mean((testing_y - predictions_lasso)^2))
rmse_lm <- sqrt(mean((testing_y - lm_prediction)^2))
rmse_lm_select <- sqrt(mean((testing_y - lm_best_select_prediction)^2))
rmse_ridge
```

```
## [1] 0.5987449
```

```
rmse_lasso
```

```
## [1] 0.5860891
```

```
rmse_lm
```

```
## [1] 0.5860069
```

```
rmse_lm_select
```

```
## [1] 0.5851154
```

From this final result, we can see that for the ridge regression, the best parameter obtained by cross validation is 0.1110505, which provides us with some regularization ability. For the lasso regression, the parameter of lambda is selected by cross validation with a value of 0.0007138406, and since this parameter is very close to 0, it can be assumed that the regularization ability of this model is relatively weak, and finally only the variable fk_acc is shrinkage to 0.

As for the results of the model evaluation in terms of RMSE, we can see that the model where the variable selection was performed manually has the lowest RMSE, except for the linear model and Lasso, although these two values are very close to each other. The worst result is the ridge, which indicates to us that the use of lasso and ridge may not be the best solution in this dataset, since lasso and ridge do not significantly improve the predictive power of the model.

```
n = 5682



cooks.distance(smallest_model2)[cooks.distance(smallest_model2) > 4 / n]
```

```
##         1047         5344         5583         5676         2082         5428
## 0.0012197667 0.0008802082 0.0014084779 0.0012762082 0.0011636036 0.0016480125
##         4767         4576         1914         4044         2244         2507
## 0.0017735937 0.0011922848 0.0018055335 0.0012856561 0.0019881938 0.0008039241
##         1313         4723         1333         3581          618         5077
## 0.0036859318 0.0010424092 0.0012161439 0.0010517779 0.0009209808 0.0011271898
##         3111           83          866         5119         4557          259
```

```
## 0.0014440253 0.0016581448 0.0013455109 0.0069860150 0.0009530936 0.0008882944
##         4590         1609         5086         5553         5681          311
## 0.0015917709 0.0009666685 0.0016481706 0.0024884780 0.0008587609 0.0018230760
##         2086         5667         5312         3043         2833         1358
## 0.0015201299 0.0011190284 0.0012323556 0.0018424784 0.0007524901 0.0007331812
##         3336         2004         5621          270         5431          601
## 0.0010276001 0.0014953814 0.0011117632 0.0007246227 0.0033955989 0.0008754310
##         5671         5294         5394         4729         5678         5041
## 0.0011227637 0.0012109122 0.0012160045 0.0007652870 0.0035114246 0.0010463652
##         5435         5587         4061         4565         1435         5663
## 0.0010429882 0.0007204882 0.0710610590 0.0010923303 0.0007176287 0.0020911319
##         1990          224         1758         2708          649          656
## 0.0041450864 0.0016099778 0.0011105918 0.0014271118 0.0015692148 0.0029494332
##          409         4680          304         4635         2761         4297
## 0.0007795368 0.0026541933 0.0007292936 0.0022097771 0.0008008592 0.0010942422
##         5597          981         4611          419         4697           45
## 0.0032694673 0.0035473353 0.0007901270 0.0013212067 0.0023818760 0.0077211404
##         1372         1379         2846         5573            1         3364
## 0.0010086613 0.0017935791 0.0007228679 0.0010125432 0.0159115189 0.0017520575
##         3327         2876          930         1565         4251          506
## 0.0607723340 0.0017193591 0.0016991761 0.0011749858 0.0010976508 0.0009431832
##         2437         1469          154         2005         3743         3675
## 0.0010305470 0.0007667458 0.0010165263 0.0012285471 0.0011843839 0.0020414587
##         5298         5397          433         1726         1144         2265
## 0.0007239848 0.0007490598 0.0009998663 0.0020200642 0.0007543395 0.0027797315
##         4588          314         2841         1178           27         2068
## 0.0010004619 0.0016511971 0.0008511571 0.0010497880 0.0081276296 0.0007708994
##         5569         3520         4929         3659         5530         2151
## 0.0026017083 0.0009714843 0.0030086096 0.0009291384 0.0014845104 0.0008062099
##         2192         2967         1895          289          167          460
## 0.0012055590 0.0008851410 0.0019789583 0.0007277662 0.0015515882 0.0021789776
##         5187         5323          264         5659          296         1797
## 0.0010809338 0.0010436669 0.0009940354 0.0015678798 0.0008106696 0.0100415116
##          655         5612         4628          365         4728         1880
## 0.0007681150 0.0052212756 0.0013390459 0.0013419674 0.0011809122 0.0014650100
##            5          815         4543            2          899         1344
## 0.0128268219 0.0009957225 0.0014650756 0.0137759722 0.0007189578 0.0012682179
##         5628         5670         5586         2405         4684         4329
## 0.0007685497 0.0008128266 0.0013647946 0.0009147359 0.0009564756 0.0012551426
##         5443         5664          978          266         1032         3707
## 0.0008375409 0.0031074496 0.0014852522 0.0008772204 0.0012082179 0.0007766640
##           20         4596         4698         5128         5135           82
## 0.0063872310 0.0029279173 0.0009949558 0.0008941893 0.0007107350 0.0013088077
##         5295         1171          861         1089          497         5292
## 0.0015085606 0.0066061653 0.0014393321 0.0007706173 0.0010996616 0.0016436888
##         5469          708         5429          522         2755         5102
## 0.0012995793 0.0011416527 0.0007997918 0.0009772053 0.0008405700 0.0007148320
##         1283           13         5584          103         5302          474
## 0.0018931709 0.0070042388 0.0022066817 0.0015544393 0.0007156047 0.0010223350
##         4707         4688           18         2336         2201          307
## 0.0026131532 0.0007783383 0.0009440264 0.0009914432 0.0007295465 0.0010265894
##         5104         2385         4653         2835         3249         3157
## 0.0008227402 0.0008580018 0.0009393926 0.0009001926 0.0031393227 0.0007656999
##          453         5438          315         2367         3673          382
```

```
## 0.0014864398 0.0020796388 0.0077094564 0.0007060013 0.0007747793 0.0016138077
##            7         4648         1151           60          218          327
## 0.0152231776 0.0061953910 0.0021444635 0.0016828824 0.0017120686 0.0009146610
##         4637         4206          408           37           78          194
## 0.0010608584 0.0011517210 0.0007103381 0.0009381725 0.0008912433 0.0008543864
##         1664         4676         3888         5629          552         2376
## 0.0008835856 0.0007321483 0.0010619914 0.0008736964 0.0007572688 0.0008314892
##         2418         5114         5143         3395         5427           42
## 0.0025289520 0.0015697920 0.0009416347 0.0022775928 0.0020330226 0.0011263816
##          260         5159           70         5483         1912         5490
## 0.0018634179 0.0010675598 0.0135383824 0.0009445051 0.0009970498 0.0035800141
##          964           12          429         2936          574         4819
## 0.0011087367 0.0043586631 0.0015273005 0.0035780323 0.0007978655 0.0023067140
##         5241          672         5065         2181          268          123
## 0.0024494534 0.0024342260 0.0008359396 0.0012020069 0.0016487421 0.0022385292
##         4678         5620         2320         1088          131         1911
## 0.0011137676 0.0040349929 0.0007902978 0.0010396290 0.0007904517 0.0007868620
##         1043         3094          910         3539         4716          938
## 0.0020836997 0.0010868623 0.0017867593 0.0010080343 0.0008450440 0.0007418777
##         4566         2928         5519         2594         5682         3549
## 0.0009369498 0.0008761865 0.0009747370 0.0007172524 0.0016146023 0.0009163976
##           32         5206         1219         5121         3388         4561
## 0.0147372558 0.0014930372 0.0012600316 0.0010812069 0.0009706317 0.0007353532
##         1198         5304         1868         4690         5276         4568
## 0.0041074429 0.0009212498 0.0009690094 0.0025122078 0.0026197319 0.0021944188
##          822         5124          882         4606         5581         4523
## 0.0009913769 0.0042552744 0.0014630768 0.0012388073 0.0015956384 0.0008132015
##         1106         3214         4562         2499          862         5572
## 0.0011343289 0.0007332361 0.0021522807 0.0007932214 0.0008667374 0.0021873619
##          144          896          984         4586         4901         4332
## 0.0017385519 0.0009078696 0.0008476492 0.0009591579 0.0012063632 0.0013202081
##         3383         5414         1752         5210         4870         2713
## 0.0009618477 0.0014386440 0.0011619406 0.0014361186 0.0013580427 0.0007378177
##         4726         3884         2029         4564         5097         2999
## 0.0007146039 0.0024481817 0.0010101576 0.0018462760 0.0022905072 0.0015080358
##          145         3316          573         1102         4124         2957
## 0.0026075104 0.0052656198 0.0007967173 0.0017931732 0.0009137192 0.0014135051
##         3170          954         4686         1095          342         1094
## 0.0012766854 0.0011004217 0.0017811487 0.0014789786 0.0021241329 0.0029277041
##         2774         3824         1334         1801          608           16
## 0.0008312752 0.0011864619 0.0010589618 0.0027983430 0.0015000374 0.0117094979
##          616         4462         3595         2722         2561         2632
## 0.0007168186 0.0011227687 0.0010588386 0.0010337634 0.0025913778 0.0017902948
##         5442          550         1573         1410         4583         5103
## 0.0008056490 0.0046575958 0.0011165682 0.0007259321 0.0007673692 0.0033403350
##         2066         4509         5305          454          424          345
## 0.0015361303 0.0012045454 0.0007089793 0.0010324249 0.0009415583 0.0007328071
##         4263          503          886         1174            9           10
## 0.0008915469 0.0007080710 0.0026018608 0.0010104720 0.0139851412 0.0093519684
##          744         2372         4627         1550         3693          704
## 0.0086038318 0.0007195539 0.0007454392 0.0008266194 0.0010426993 0.0008229932
##         4068         5492         1038         5506         4714         5437
## 0.0024603265 0.0014184121 0.0483592131 0.0010109925 0.0018261394 0.0012084946
##         5072          775            3         4570          546         4694
```

```
## 0.0013551496 0.0007305942 0.0118368249 0.0015175398 0.0007787679 0.0008639235
##         4841          653         2595          818         4642         1769
## 0.0021057198 0.0012730797 0.0007172524 0.0012894524 0.0023397270 0.0007615416
##         2902         5098         1173         5582         1040         4657
## 0.0008504161 0.0020856389 0.0017180203 0.0014092252 0.0009673396 0.0016120005
##         3044          521          908         2050          130         4712
## 0.0015988061 0.0007860259 0.0014452259 0.0009324191 0.0011827925 0.0011960274
##          146         2114         3179         2603         4739         1819
## 0.0009156440 0.0007898809 0.0018604604 0.0020904641 0.0007046429 0.0008603240
##         3533         4018          838         5092         1343
## 0.0036649793 0.0017616452 0.0011257104 0.0012428803 0.0011053604
```

```
finalmodel_noinfl = project_data_1[!(cooks.distance(smallest_model2) > 4 / n),]
dim(finalmodel_noinfl)
```

```
## [1] 5192   40
```

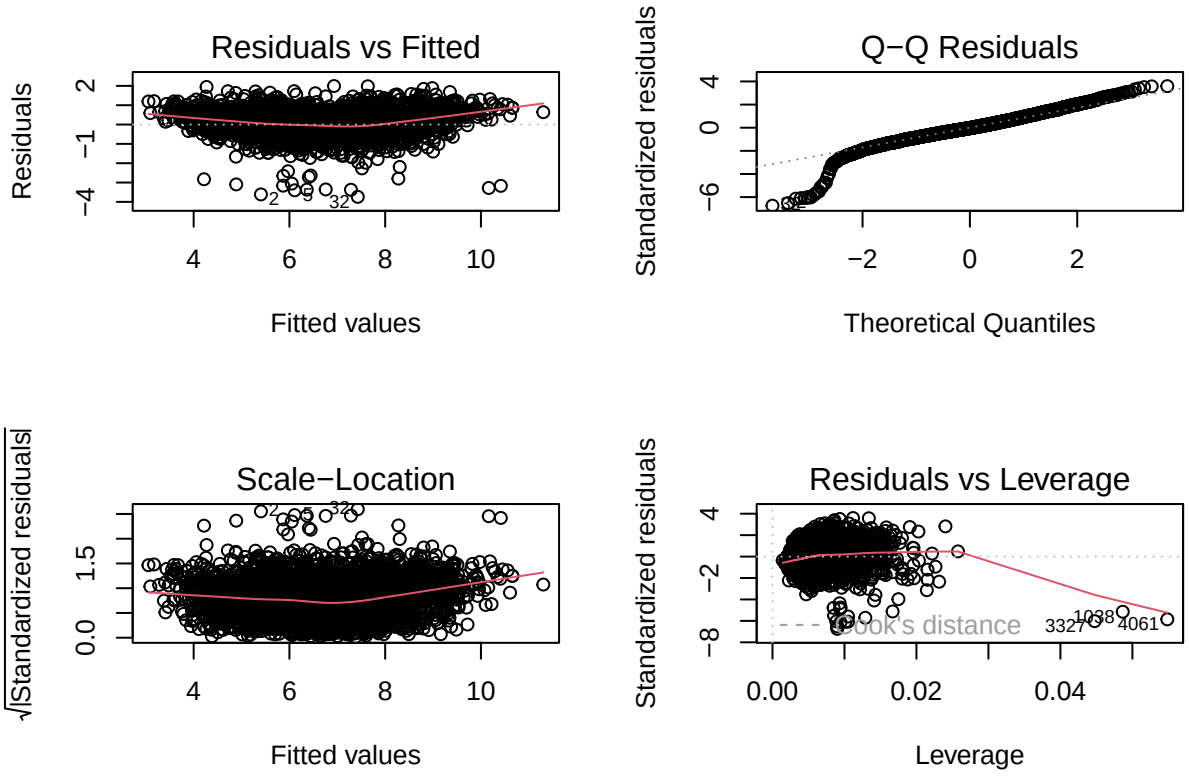
```
#After looking through our observations to see that 346 observations have a cooks distance greater
#than 4/(number of observations)


final_model_noinfl = lm(log(value) ~. -player -country,
                        data = finalmodel_noinfl)
summary(final_model_noinfl)
```

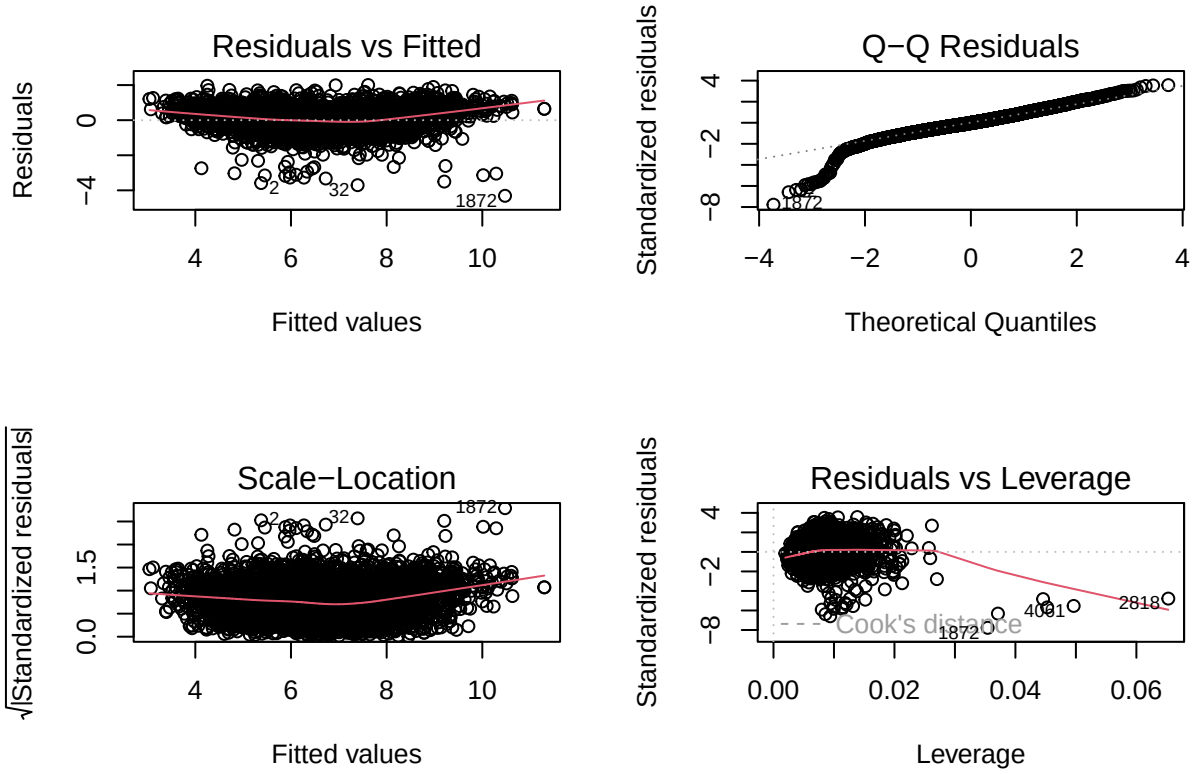
```
##
## Call:
## lm(formula = log(value) ~ . - player - country, data = finalmodel_noinfl)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3116 -0.3192 -0.0116  0.3408  2.0027
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -3.1697231  0.4152316  -7.634 2.70e-14 ***
## height         -0.0005521  0.0022931  -0.241 0.809730
## weight          0.0020797  0.0019383   1.073 0.283328
## age            -0.0548685  0.0021980 -24.963  < 2e-16 ***
## ball_control    0.0284881  0.0022806  12.492  < 2e-16 ***
## dribbling       0.0039181  0.0018517   2.116 0.034396 *
## slide_tackle   -0.0011821  0.0019953  -0.592 0.553590
## stand_tackle    0.0073025  0.0020816   3.508 0.000455 ***
## aggression     -0.0009500  0.0009580  -0.992 0.321417
## reactions       0.0693707  0.0016282  42.607  < 2e-16 ***
## att_position   -0.0078208  0.0014024  -5.577 2.58e-08 ***
## interceptions  -0.0038414  0.0014450  -2.658 0.007874 **
## vision         -0.0040225  0.0012759  -3.153 0.001628 **
## composure       0.0115331  0.0013814   8.349  < 2e-16 ***
## crossing        0.0104726  0.0011941   8.770  < 2e-16 ***
## short_pass      0.0236677  0.0022524  10.508  < 2e-16 ***
## long_pass      -0.0068016  0.0015871  -4.285 1.86e-05 ***
## acceleration    0.0069685  0.0016496   4.224 2.44e-05 ***
## stamina         0.0084762  0.0009783   8.664  < 2e-16 ***
## strength        0.0065058  0.0011693   5.564 2.78e-08 ***
```

```
## balance         -0.0016462  0.0011924  -1.381 0.167488
## sprint_speed     0.0052117  0.0014312   3.641 0.000274 ***
## agility         -0.0023564  0.0013004  -1.812 0.070031 .
## jumping         -0.0011371  0.0008509  -1.336 0.181491
## heading          0.0141935  0.0012088  11.742  < 2e-16 ***
## shot_power       0.0111566  0.0013414   8.317  < 2e-16 ***
## finishing        0.0052076  0.0015022   3.467 0.000531 ***
## long_shots      -0.0057217  0.0014294  -4.003 6.35e-05 ***
## curve            0.0005394  0.0012742   0.423 0.672054
## fk_acc          -0.0005389  0.0011488  -0.469 0.639033
## penalties       -0.0041145  0.0011956  -3.441 0.000583 ***
## volleys         -0.0006824  0.0012321  -0.554 0.579701
## gk_positioning   0.0131369  0.0024818   5.293 1.25e-07 ***
## gk_diving        0.0126023  0.0024956   5.050 4.58e-07 ***
## gk_handling      0.0061084  0.0025481   2.397 0.016553 *
## gk_kicking       0.0002004  0.0024192   0.083 0.933975
## gk_reflexes      0.0126338  0.0024657   5.124 3.10e-07 ***
## goal_keeper      0.6944570  0.1783696   3.893 0.000100 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5646 on 5154 degrees of freedom
## Multiple R-squared:  0.8246, Adjusted R-squared:  0.8233
## F-statistic: 654.8 on 37 and 5154 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2,2))
plot(smallest_model2)
```



```r
plot(final_model_noinfl)
```



After looking through our observations to see that 346 observations have a cooks distance greater than 4/(number of observations). After this, we refitted a new model with 5,336 observations preformed diagnostics. From this we can see from the residual vs fitted plot that our residuals are much more consistent than before, following an almost flat line.

```r
legendary = data.frame(player = c("Ronaldinho", "Thierry Henry", "John Terry", "Petr Cech"),
                country=c("Brazil", "France", "England", "Czech Republic"),
                height=c(180, 188, 187, 195),
                weight=c(80, 83, 90, 92),
                age=c(27, 30, 26, 25),
                ball_control=c(97, 91, 65, 24),
                dribbling=c(97, 92, 41, 20),
                slide_tackle=c(28, 21, 93, 28),
                stand_tackle=c(28, 21, 93, 28),
                aggression=c(66, 40, 93, 67),
                reactions=c(91, 92, 85, 83),
                att_position=c(80, 80, 60, 40),
                interceptions=c(28, 21, 93, 28),
                vision=c(84, 87, 80, 57),
                composure=c(83, 92, 85, 77),
                crossing=c(83, 76, 41, 20),
                short_pass=c(90, 86, 65, 25),
                long_pass=c(85, 67, 61, 84),
                acceleration=c(89, 94, 69, 53),
                stamina=c(80, 83, 87, 48),
                strength=c(76, 78, 93, 74),
                balance=c(84, 87, 80, 57),
```

```
                 sprint_speed=c(91, 93, 71, 54),
                 agility=c(84, 87, 80, 57),
                 jumping=c(84, 87, 80, 57),
                 heading=c(66, 65, 94, 20),
                 shot_power=c(84, 86, 58, 21),
                 finishing=c(90, 95, 36, 20),
                 long_shots=c(93, 86, 33, 20),
                 curve=c(90, 95, 36, 20),
                 fk_acc=c(93, 89, 29, 19),
                 penalties=c(93, 89, 29, 19),
                 volleys=c(90, 95, 36, 20),
                 gk_positioning=c(22, 21, 23, 88),
                 gk_diving=c(12, 14, 7, 91),
                 gk_handling=c(22, 21, 23, 90),
                 gk_kicking=c(80, 80, 80, 80),
                 gk_reflexes=c(22, 21, 23, 92),
                 goal_keeper=c(0, 0, 0, 1))
# Ensure 'player' is a factor with the same levels as in the training data
legendary$player <- factor(legendary$player, levels = levels(final_model_noinfl$model$player))

# Make predictions
predictions <- predict(final_model_noinfl, newdata = legendary)

# Display predictions of log values and real player values
predictions
```

```
##         1         2         3         4
## 11.356653 11.281369  9.721689  9.026079
```

```
exp(predictions)
```

```
##         1         2         3         4
## 85532.607 79329.795 16675.384  8317.187
```

We first researched a website to find the attributes of older, legendary players based on a older version of FIFA. I then filled in the values based on the attributes given on the website. There were missing attributes on the website that we needed for our model so the following rules were applied to fill in missing values. slide_tackle, stand_tackle and interceptions were filled using the value of tackling on the website. att_position was filled based on the players position, attackers got a 80, defenders got a 60 and goalkeepers got a 40. Vision, balance, agility and jumping were all the average of the physical attributes given on the website which were Acceleration, Stamina, Strength and Sprint Speed. Curve and Volleys were filled in based on the finishing attribute of the player and finally penalties were filled using the fk_acc attribute of the player. After filling the data frame out we predicted the value of the player in dollars using the optimal model we identified and the log values were printed out as "predictions". To find the player value in dollars that the model predicted we applied the inverse log function and we have that Ronaldinho would be valued at 85k dollars, Thierry Henry would be valued at 79k dollars, John Terry would be valued at 16k dollars and Petr Cech would be valued at 8k dollars. While these are high numbers they are still lower than the player values of the highest valued player nowadays like Kylian Mbappe valued at 153k dollars and Erling Haaland valued at 123k dollars.