test new code

Brayden Adams

2024-12-11

```r
# Load necessary library
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.3.2

## Loaded glmnet 4.1-8
```

```r
library(ggplot2)

# Define player stats (including Furkan Korkmaz)
players <- data.frame(
  name = c(
    "Nikola Jokic", "Luka Doncic", "Joel Embiid", "Giannis Antetokounmpo", "Shai Gilgeous-Alexander",
    "Anthony Davis", "LeBron James", "Kevin Durant", "Jayson Tatum", "Stephen Curry", "Killian Hayes",
    "Patrick Beverley", "Jay Huff", "Nicolas Claxton", "Desmond Bane", "Tobias Harris", "Paolo Banchero
    "Myles Turner", "Austin Reaves", "D'Angelo Russell", "Ja Morant", "Furkan Korkmaz", "Kyrie Irving",
    "Lamelo Ball", "Jaylen Brown", "Devin Booker", "James Harden", "Damian Lillard", "Jimmy Butler",
    "Zion Williamson", "De'Aaron Fox", "Jrue Holiday", "Bam Adebayo", "Jaren Jackson Jr.", "Donovan Mit
    "Klay Thompson", "Chris Paul", "Karl-Anthony Towns", "Brandon Ingram", "DeMar DeRozan", "Trae Young
    "Bradley Beal", "Pascal Siakam", "CJ McCollum", "Tyrese Haliburton", "Kawhi Leonard", "Paul George"
    "Fred VanVleet", "Kristaps Porzingis", "Rudy Gobert", "Precious Achiuwa", "Steven Adams", "Ochai Ag
    "Santi Aldama", "Trey Alexander", "Nickeil Alexander-Walker", "Grayson Allen", "Jarrett Allen", "Jo
    "Kyle Anderson", "Cole Anthony", "OG Anunoby", "Deni Avdija", "Deandre Ayton", "Marvin Bagley III",
    "Patrick Baldwin Jr.", "Lonzo Ball", "Mo Bamba", "Dalano Banton", "Dominick Barlow", "Harrison Barn
    "Scottie Barnes", "RJ Barrett", "Charles Bassey", "Emoni Bates", "Jamison Battle", "Nicolas Batum",
    "Malik Beasley", "MarJon Beauchamp", "Reece Beekman", "Saddiq Bey", "Goga Bitadze", "Anthony Black"
    "Bogdan Bogdanović", "Bojan Bogdanović", "Bol Bol", "Adem Bona", "Brandon Boston Jr.", "Chris Bouch
    "Malaki Branham", "Christian Braun", "Jalen Bridges", "Mikal Bridges", "Michael Porter Jr.", "Darius
  ),
  points_per_game = c(26.4, 33.9, 34.7, 30.4, 30.1, 24.7, 25.7, 27.1, 28.4, 26.4, 6.9, 6.2, 3.5, 12.6, 
  defensive_rating = c(107, 110, 107, 102, 108, 104, 106, 109, 107, 111, 110, 110, 103, 101, 106, 105, 
  assists_per_game = c(9.0, 9.8, 5.6, 6.5, 6.2, 3.5, 8.3, 5.0, 5.6, 5.1, 4.9, 2.9, 1.0, 1.5, 4.4, 2.5, 
  per = c(32.1, 23.5, 28.3, 29.5, 27.8, 26.4, 25.7, 27.1, 26.9, 28.5, 10.5, 12.2, 15.0, 20.3, 19.8, 17.5
  win_shares_per_48 = c(0.301, 0.250, 0.270, 0.280, 0.240, 0.220, 0.230, 0.250, 0.260, 0.270, 0.050, 0.0
  bpm = c(8.5, 7.2, 8.0, 7.8, 6.5, 6.9, 7.5, 7.3, 7.0, 7.6, 2.5, 2.8, 1.0, 4.5, 5.0, 4.0, 3.8, 4.2, 3.2
  vorp = c(7.0, 6.5, 6.8, 6.7, 5.5, 5.8, 6.2, 6.0, 5.9, 6.3, 1.5, 1.8, 0.5, 3.5, 4.0, 3.0, 3.5, 4.0, 3.0
```

```r
  nba_2k25_rating = c( 98, 96, 95, 98, 96, 96, 95, 95, 96, 95, 79, 78, 70, 81, 83, 84, 89, 84, 81, 81, 9
```

```r
# Adjusted normalization function to scale to a range of 67 to 99
adjusted_norm <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)) * (99 - 67) + 67)  # Scale to a range that ensures the lowes
}

# Apply adjusted normalization to each criterion
players$scoring <- adjusted_norm(players$points_per_game)
players$defense <- adjusted_norm(max(players$defensive_rating) - players$defensive_rating)  # Invert be
players$playmaking <- adjusted_norm(players$assists_per_game)
players$efficiency <- adjusted_norm(players$per)
players$impact <- adjusted_norm(players$win_shares_per_48)
players$bpm_norm <- adjusted_norm(players$bpm)
players$vorp_norm <- adjusted_norm(players$vorp)

# Calculate final rating
players$final_rating_formula <- rowMeans(players[, c("scoring", "defense", "playmaking", "efficiency",

# Prepare data for ridge regression
x <- as.matrix(players[, c("points_per_game", "defensive_rating", "assists_per_game", "per", "win_share
y <- players$nba_2k25_rating

# Fit ridge regression model
ridge_model <- cv.glmnet(x, y, alpha = 0)

# Predict ratings using the ridge regression model
players$predicted_rating_ridge <- predict(ridge_model, s = "lambda.min", newx = x)

# Calculate correlation coefficient between predicted_rating and 2K rating
correlation_coefficient_formula <- cor(players$final_rating_formula, players$nba_2k25_rating)
correlation_coefficient_ridge <- cor(players$predicted_rating_ridge, players$nba_2k25_rating)

# Rank players by final_rating in descending order
ranked_players_formula <- players[order(-players$final_rating_formula), ]
ranked_players_ridge <- players[order(-players$predicted_rating_ridge), ]

# Display final ratings and correlation coefficients
print("Ranked Players by Formula-Based Ratings:")
```

```
## [1] "Ranked Players by Formula-Based Ratings:"
```

```r
print(ranked_players_formula[, c("name", "final_rating_formula", "nba_2k25_rating")])
```

```
##                       name final_rating_formula nba_2k25_rating
## 4     Giannis Antetokounmpo             93.42214              98
## 1              Nikola Jokic             92.63102              98
## 3               Joel Embiid             90.19681              95
## 28           Damian Lillard             89.18151              89
```

```
## 2              Luka Doncic          89.09793           96
## 7             LeBron James          88.90545           95
## 26            Devin Booker          88.60244           93
## 9             Jayson Tatum          88.23471           96
## 5   Shai Gilgeous-Alexander         88.15015           96
## 31             De'Aaron Fox         88.12615           88
## 41               Trae Young         87.98255           89
## 24              Lamelo Ball         87.16134           87
## 46            Kawhi Leonard         86.82034           92
## 27             James Harden         86.69002           86
## 6            Anthony Davis          86.68169           96
## 48           Fred VanVleet          86.54066           84
## 21               Ja Morant          86.50682           91
## 49      Kristaps Porzingis          86.33323           87
## 8             Kevin Durant          86.21685           95
## 23            Kyrie Irving          86.13804           92
## 35        Donovan Mitchell          86.06529           93
## 47             Paul George          85.95935           89
## 34        Jaren Jackson Jr.         85.95398           87
## 10           Stephen Curry          85.89970           95
## 42            Bradley Beal          85.65097           85
## 39          Brandon Ingram          85.43328           85
## 45        Tyrese Haliburton         85.05494           90
## 38      Karl-Anthony Towns          85.04885           92
## 44             CJ McCollum          84.99736           84
## 43           Pascal Siakam          84.54729           88
## 32            Jrue Holiday          84.31051           85
## 30         Zion Williamson          84.15241           88
## 29            Jimmy Butler          83.99733           89
## 25            Jaylen Brown          83.96912           92
## 33             Bam Adebayo          83.91853           88
## 40           DeMar DeRozan          83.74544           87
## 15            Desmond Bane          82.98388           83
## 37              Chris Paul          81.66016           81
## 14         Nicolas Claxton          81.04670           81
## 50             Rudy Gobert          80.92564           85
## 18            Myles Turner          80.77287           84
## 16           Tobias Harris          80.41549           84
## 51        Precious Achiuwa          80.25339           78
## 36           Klay Thompson          80.12863           81
## 20        D'Angelo Russell          79.49971           81
## 78           Malik Beasley          79.19853           81
## 76          Jamison Battle          78.86408           70
## 17          Paolo Banchero          78.65458           89
## 82            Goga Bitadze          78.55908           78
## 86                 Bol Bol          77.76913           77
## 85        Bojan Bogdanović          77.60997           81
## 62              OG Anunoby          77.55204           84
## 72          Scottie Barnes          76.99730           85
## 90          Malaki Branham          76.95581           75
## 94      Michael Porter Jr.          76.89832           83
## 55          Trey Alexander          76.83651           69
## 73              RJ Barrett          76.68937           81
## 59           Jose Alvarado          76.68119           75
```

```
## 68                   Mo Bamba            76.67217              78
## 96              Julius Randle            76.66413              85
## 64              Deandre Ayton            76.62824              82
## 93              Mikal Bridges            76.57977              84
## 80              Reece Beekman            76.56685              73
## 63                Deni Avdija            76.54539              78
## 65          Marvin Bagley III            76.46339              77
## 95             Darius Garland            76.44844              82
## 69              Dalano Banton            76.26317              73
## 70            Dominick Barlow            76.26012              70
## 19              Austin Reaves            76.24705              81
## 87                  Adem Bona            76.14086              73
## 92              Jalen Bridges            75.97454              73
## 75                Emoni Bates            75.88134              72
## 60              Kyle Anderson            75.73017              78
## 58              Jarrett Allen            75.70562              84
## 66        Patrick Baldwin Jr.            75.65015              74
## 81                 Saddiq Bey            75.58056              81
## 67                 Lonzo Ball            75.52636              81
## 56 Nickeil Alexander-Walker            75.43538              80
## 79           MarJon Beauchamp            75.36203              75
## 77              Nicolas Batum            75.34937              79
## 83              Anthony Black            75.33512              70
## 54               Santi Aldama            75.29672              70
## 71            Harrison Barnes            75.29417              80
## 74             Charles Bassey            75.11324              76
## 88         Brandon Boston Jr.            75.10804              78
## 89              Chris Boucher            75.04008              79
## 84          Bogdan Bogdanović            74.89024              82
## 13                   Jay Huff            74.84460              70
## 91            Christian Braun            74.55225              74
## 61               Cole Anthony            73.72611              80
## 52               Steven Adams            73.60784              79
## 57              Grayson Allen            73.21162              81
## 11              Killian Hayes            71.40801              79
## 53               Ochai Agbaji            71.30726              72
## 12           Patrick Beverley            70.46200              78
## 22             Furkan Korkmaz            69.17858              76
```

```r
print(paste("The correlation coefficient between the formula ratings and the actual ratings is", correla
```

```
## [1] "The correlation coefficient between the formula ratings and the actual ratings is 0.82833139640
```

```r
print("\nRanked Players by Ridge Regression-Based Ratings:")
```

```
## [1] "\nRanked Players by Ridge Regression-Based Ratings:"
```

```r
print(ranked_players_ridge[, c("name", "predicted_rating_ridge", "nba_2k25_rating")])
```

```
##                        name lambda.min nba_2k25_rating
## 1              Nikola Jokic   96.96814              98
## 3               Joel Embiid   96.65467              95
```

```
## 4     Giannis Antetokounmpo   96.37995               98
## 2               Luka Doncic   95.06277               96
## 5   Shai Gilgeous-Alexander   94.73450               96
## 41               Trae Young   93.91674               89
## 9              Jayson Tatum   93.60606               96
## 10            Stephen Curry   93.11799               95
## 28           Damian Lillard   93.03152               89
## 8              Kevin Durant   92.69225               95
## 7              LeBron James   92.64872               95
## 26             Devin Booker   92.10433               93
## 6             Anthony Davis   90.75673               96
## 31             De'Aaron Fox   90.43418               88
## 24              Lamelo Ball   89.60743               87
## 35          Donovan Mitchell   89.43219               93
## 47              Paul George   89.20339               89
## 46            Kawhi Leonard   89.06822               92
## 21                Ja Morant   89.02391               91
## 23             Kyrie Irving   88.91505               92
## 27             James Harden   88.80899               86
## 42             Bradley Beal   88.53617               85
## 38       Karl-Anthony Towns   88.16731               92
## 39           Brandon Ingram   87.53539               85
## 30          Zion Williamson   87.40826               88
## 29             Jimmy Butler   87.17109               89
## 49       Kristaps Porzingis   87.14437               87
## 44              CJ McCollum   87.11893               84
## 15             Desmond Bane   86.87951               83
## 25             Jaylen Brown   86.80900               92
## 34        Jaren Jackson Jr.   86.75611               87
## 45        Tyrese Haliburton   86.55655               90
## 32             Jrue Holiday   86.49924               85
## 48            Fred VanVleet   86.34334               84
## 40            DeMar DeRozan   86.29336               87
## 43            Pascal Siakam   85.33070               88
## 33              Bam Adebayo   84.73239               88
## 36            Klay Thompson   84.09327               81
## 16            Tobias Harris   83.80179               84
## 37               Chris Paul   83.34550               81
## 17           Paolo Banchero   83.18781               89
## 20          D'Angelo Russell   83.17475               81
## 50              Rudy Gobert   82.27941               85
## 14          Nicolas Claxton   82.27120               81
## 18             Myles Turner   81.96024               84
## 76           Jamison Battle   81.07083               70
## 51         Precious Achiuwa   81.05691               78
## 87                Adem Bona   80.48116               73
## 73               RJ Barrett   80.42650               81
## 78            Malik Beasley   80.31143               81
## 82             Goga Bitadze   80.26023               78
## 94       Michael Porter Jr.   79.95783               83
## 66      Patrick Baldwin Jr.   79.79228               74
## 86                  Bol Bol   79.76697               77
## 63              Deni Avdija   79.35938               78
## 65        Marvin Bagley III   79.26391               77
```

```
## 19           Austin Reaves   79.08775              81
## 59           Jose Alvarado   79.06671              75
## 62              OG Anunoby   78.92773              84
## 96           Julius Randle   78.76345              85
## 67              Lonzo Ball   78.76007              81
## 93           Mikal Bridges   78.51776              84
## 64           Deandre Ayton   78.44606              82
## 72          Scottie Barnes   78.36658              85
## 85        Bojan Bogdanović   78.34528              81
## 77           Nicolas Batum   78.30929              79
## 70         Dominick Barlow   78.25331              70
## 68                Mo Bamba   78.17727              78
## 91         Christian Braun   77.98014              74
## 74          Charles Bassey   77.95387              76
## 58           Jarrett Allen   77.94099              84
## 61            Cole Anthony   77.86624              80
## 81              Saddiq Bey   77.82135              81
## 60           Kyle Anderson   77.75213              78
## 75             Emoni Bates   77.59009              72
## 80           Reece Beekman   77.52126              73
## 95          Darius Garland   77.44024              82
## 71         Harrison Barnes   77.39542              80
## 69           Dalano Banton   77.26130              73
## 92           Jalen Bridges   77.22749              73
## 52            Steven Adams   77.21458              79
## 79        MarJon Beauchamp   77.00140              75
## 89           Chris Boucher   76.64460              79
## 90          Malaki Branham   76.64228              75
## 54            Santi Aldama   76.48982              70
## 88      Brandon Boston Jr.   76.32503              78
## 55          Trey Alexander   76.25047              69
## 83           Anthony Black   76.22798              70
## 84       Bogdan Bogdanović   76.19674              82
## 56 Nickeil Alexander-Walker   76.17298              80
## 57           Grayson Allen   76.05077              81
## 13                Jay Huff   75.96447              70
## 11           Killian Hayes   75.30717              79
## 53            Ochai Agbaji   75.20174              72
## 12        Patrick Beverley   74.92460              78
## 22          Furkan Korkmaz   74.30278              76
```

```
print("The correlation coefficient between the ridge regression ratings and the actual ratings is")
```

```
## [1] "The correlation coefficient between the ridge regression ratings and the actual ratings is"
```

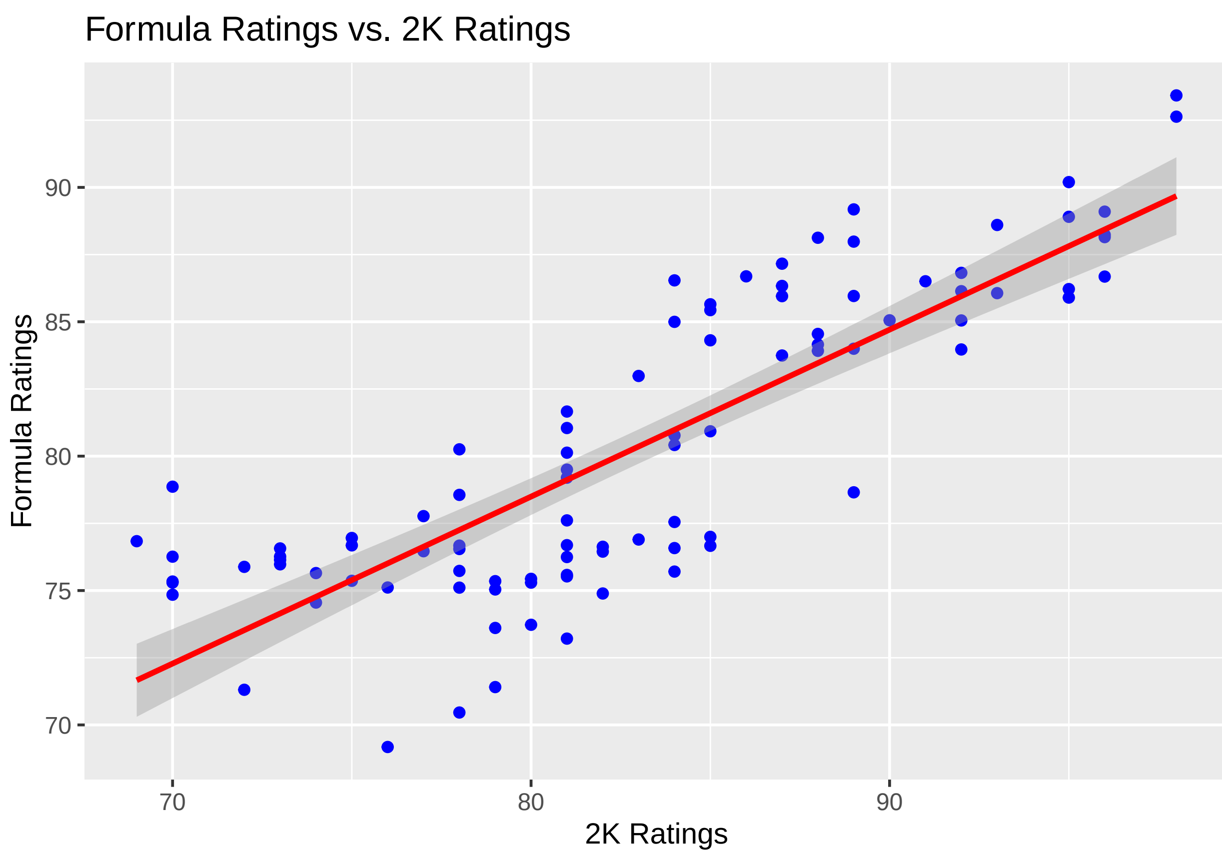
```
correlation_coefficient_ridge
```

```
##                   [,1]
## lambda.min 0.8641309
```

```
library(ggplot2)
ggplot(players, aes(x = nba_2k25_rating, y = final_rating_formula)) +
```

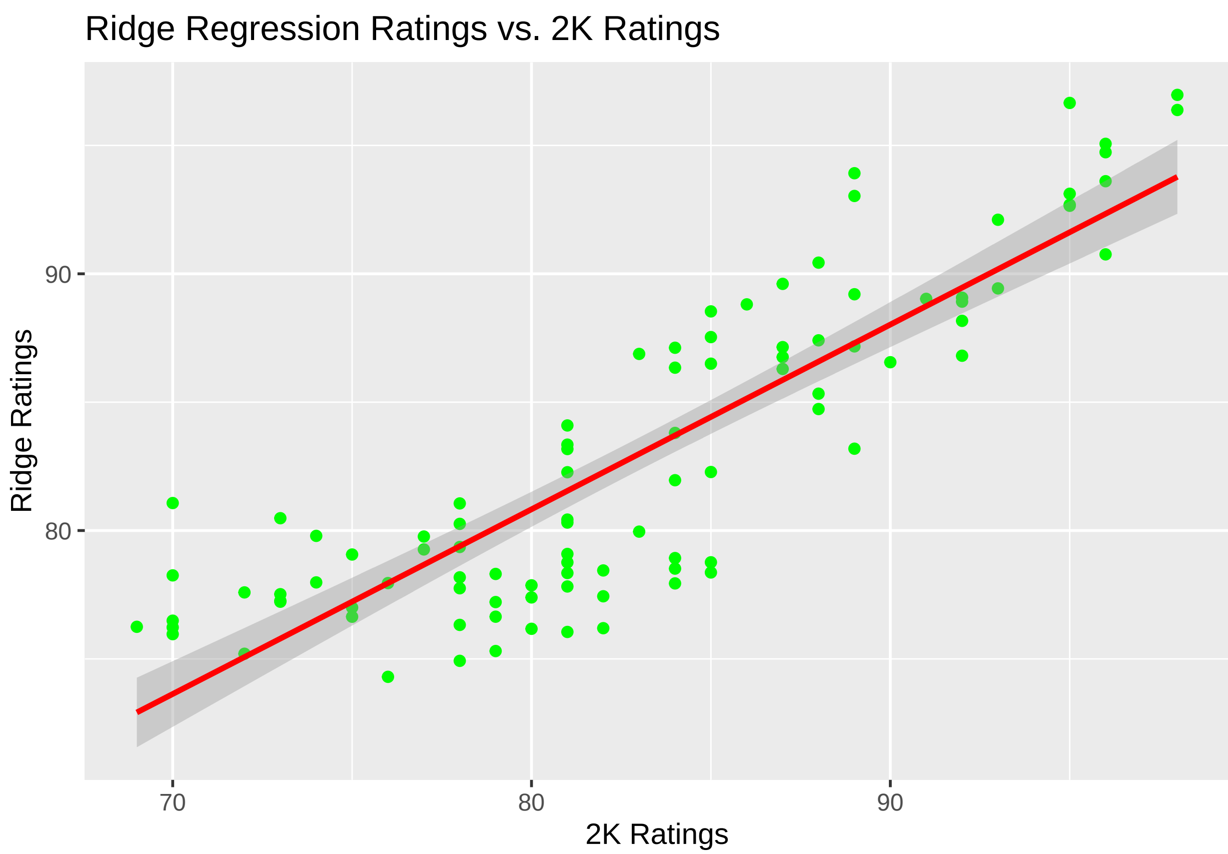
```r
  geom_point(color = 'blue') +
  geom_smooth(method = 'lm', color = 'red') +
  labs(title = 'Formula Ratings vs. 2K Ratings', x = '2K Ratings', y = 'Formula Ratings')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
ggplot(players, aes(x = nba_2k25_rating, y = predicted_rating_ridge)) +
  geom_point(color = 'green') +
  geom_smooth(method = 'lm', color = 'red') +
  labs(title = 'Ridge Regression Ratings vs. 2K Ratings', x = '2K Ratings', y = 'Ridge Ratings')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
write.csv(ranked_players_formula, "formula_based_rankings.csv")
write.csv(ranked_players_ridge, "ridge_based_rankings.csv")
coef(ridge_model, s = "lambda.min")
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)        73.68101935
## points_per_game     0.40555879
## defensive_rating   -0.07953497
## assists_per_game    0.39953921
## per                 0.48401845
## win_shares_per_48  13.67853254
## bpm                 0.46498870
## vorp               -0.87312530
```

```r
# Hyperparameter tuning for lambda in ridge regression
lambda_grid <- 10^seq(3, -3, by = -1)
ridge_model_tuned <- cv.glmnet(x, y, alpha = 0, lambda = lambda_grid)
best_lambda <- ridge_model_tuned$lambda.min
print(paste("Best lambda:", best_lambda))
```

```
## [1] "Best lambda: 1"
```

```r
# Feature importance based on ridge regression coefficients
coefficients <- coef(ridge_model, s = "lambda.min")
coefficients_df <- data.frame(
  feature = rownames(coefficients),
  importance = as.vector(coefficients)
)
coefficients_df <- coefficients_df[-1,]  # Remove the intercept
coefficients_df <- coefficients_df[order(abs(coefficients_df$importance), decreasing = TRUE), ]

print("Feature Importance from Ridge Regression:")
```

```
## [1] "Feature Importance from Ridge Regression:"
```

```r
print(coefficients_df)
```

```
##             feature  importance
## 6 win_shares_per_48 13.67853254
## 8              vorp -0.87312530
## 5               per  0.48401845
## 7               bpm  0.46498870
## 2   points_per_game  0.40555879
## 4  assists_per_game  0.39953921
## 3  defensive_rating -0.07953497
```

```r
# Input player names and rank based on formula or ridge regression
custom_player_names <- c("Nikola Jokic", "Stephen Curry", "Furkan Korkmaz")
custom_players <- players[players$name %in% custom_player_names, ]
custom_players_formula <- custom_players[order(-custom_players$final_rating_formula), ]
custom_players_ridge <- custom_players[order(-custom_players$predicted_rating_ridge), ]

print("Custom Player Rankings based on Formula:")
```

```
## [1] "Custom Player Rankings based on Formula:"
```

```r
print(custom_players_formula[, c("name", "final_rating_formula")])
```

```
##              name final_rating_formula
## 1    Nikola Jokic             92.63102
## 10  Stephen Curry             85.89970
## 22 Furkan Korkmaz             69.17858
```

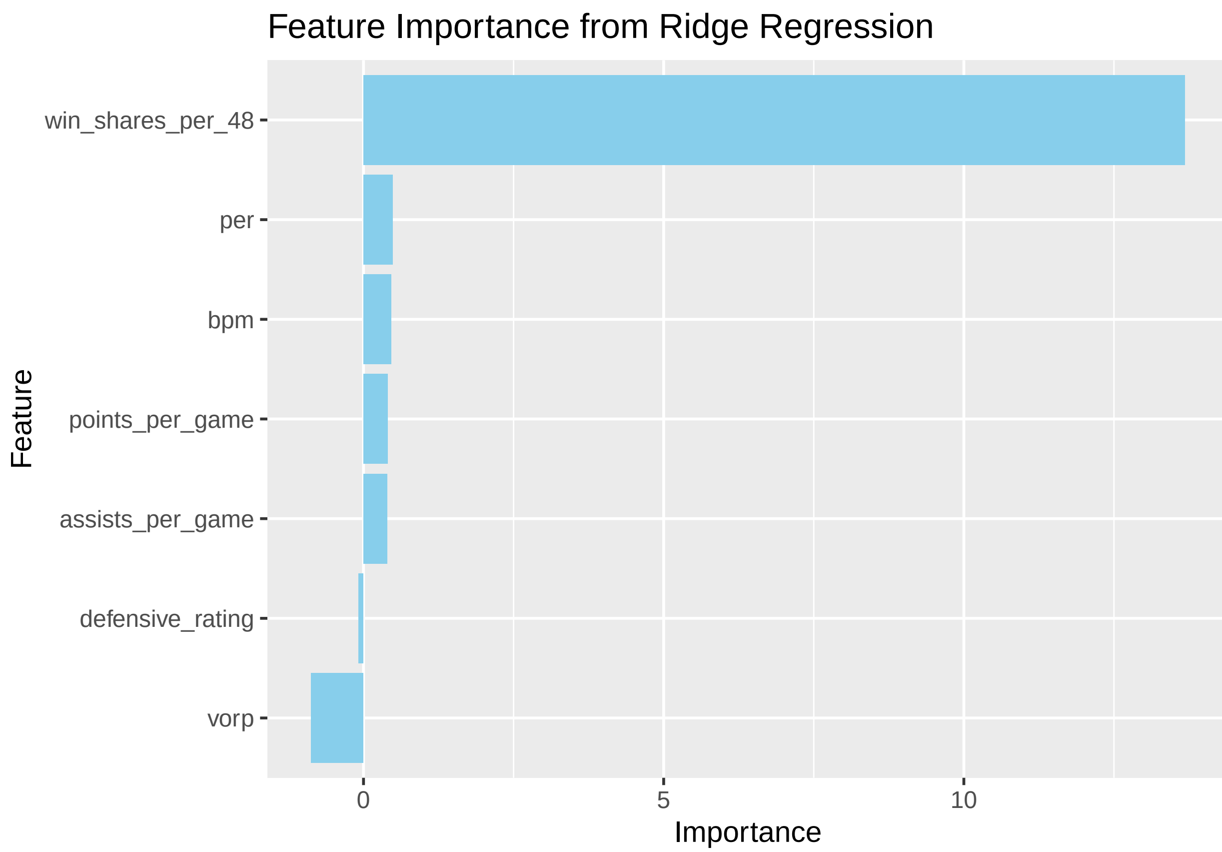
```r
print("Custom Player Rankings based on Ridge Regression:")
```

```
## [1] "Custom Player Rankings based on Ridge Regression:"
```

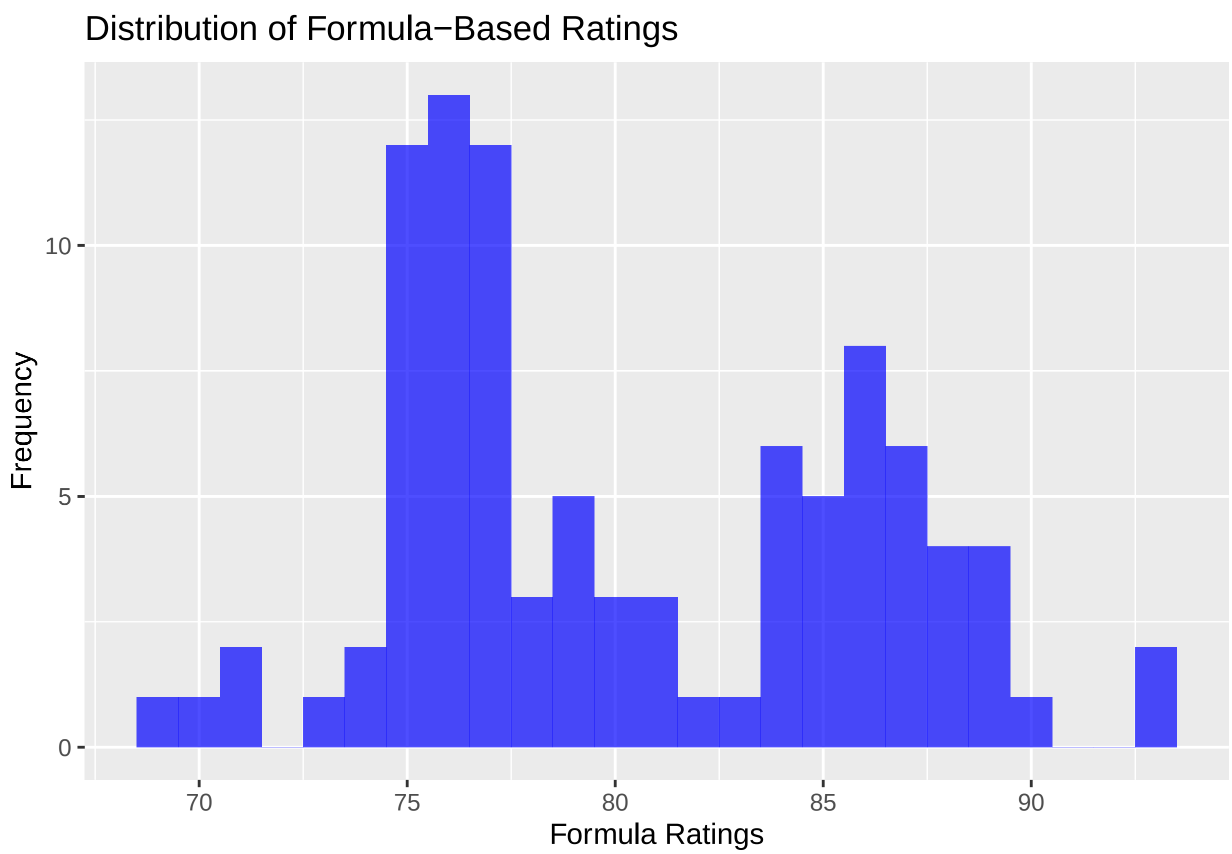
```r
print(custom_players_ridge[, c("name", "predicted_rating_ridge")])
```

```
##              name lambda.min
## 1    Nikola Jokic   96.96814
## 10  Stephen Curry   93.11799
## 22 Furkan Korkmaz   74.30278
```

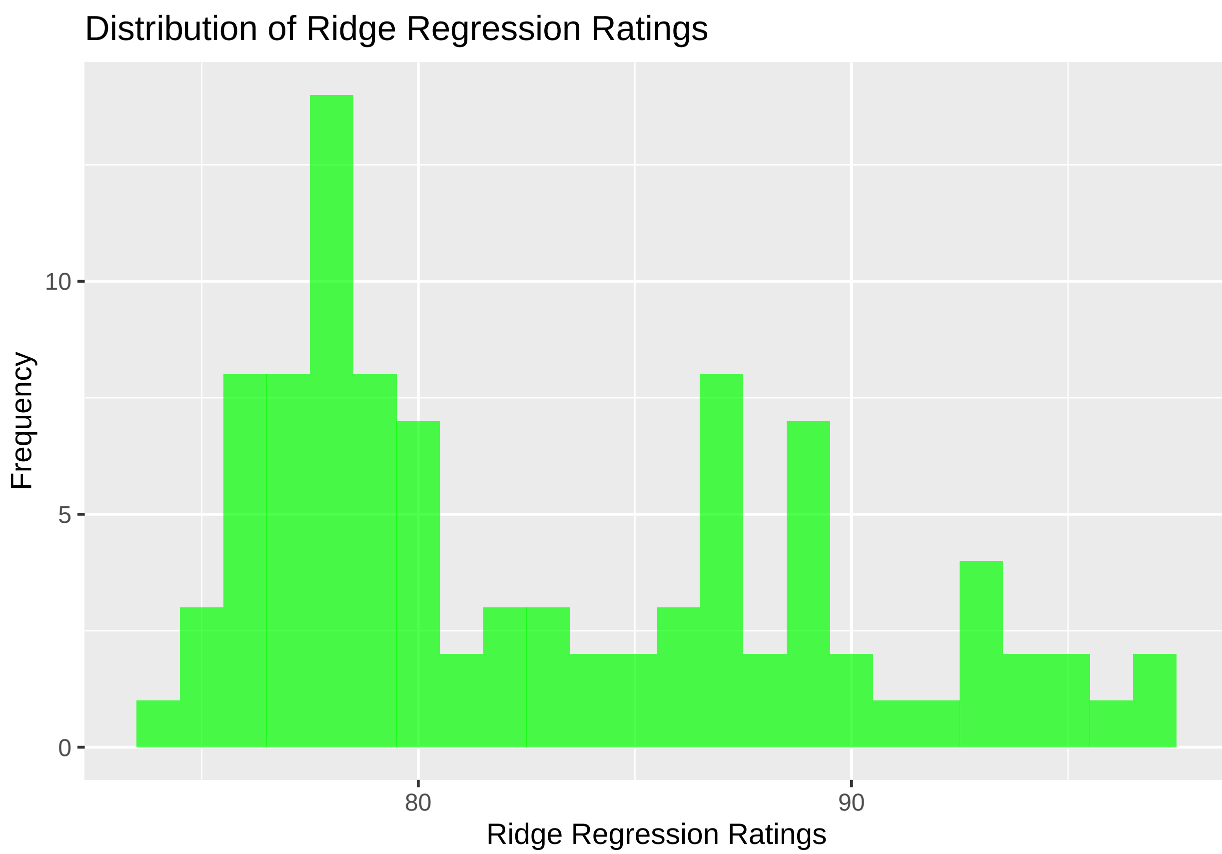
```r
# Save feature importance to CSV
write.csv(coefficients_df, "feature_importance.csv")

# Save custom player rankings
write.csv(custom_players_formula, "custom_player_formula_rankings.csv")
write.csv(custom_players_ridge, "custom_player_ridge_rankings.csv")

# Bar plot of feature importance
ggplot(coefficients_df, aes(x = reorder(feature, importance), y = importance)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Feature Importance from Ridge Regression", x = "Feature", y = "Importance")
```




```r
# Distribution plot of the ratings
ggplot(players, aes(x = final_rating_formula)) +
  geom_histogram(binwidth = 1, fill = "blue", alpha = 0.7) +
  labs(title = "Distribution of Formula-Based Ratings", x = "Formula Ratings", y = "Frequency")
```




```r
ggplot(players, aes(x = predicted_rating_ridge)) +
  geom_histogram(binwidth = 1, fill = "green", alpha = 0.7) +
  labs(title = "Distribution of Ridge Regression Ratings", x = "Ridge Regression Ratings", y = "Frequenc
```




```r
# Calculate MAE and RMSE for both models
mae_formula <- mean(abs(players$final_rating_formula - players$rating_2k))
rmse_formula <- sqrt(mean((players$final_rating_formula - players$rating_2k)^2))

mae_ridge <- mean(abs(players$predicted_rating_ridge - players$rating_2k))
rmse_ridge <- sqrt(mean((players$predicted_rating_ridge - players$rating_2k)^2))

print(paste("MAE for formula-based ratings:", mae_formula))
```

```
## [1] "MAE for formula-based ratings: NaN"
```

```r
print(paste("RMSE for formula-based ratings:", rmse_formula))
```

```
## [1] "RMSE for formula-based ratings: NaN"
```

```r
print(paste("MAE for ridge regression ratings:", mae_ridge))
```

```
## [1] "MAE for ridge regression ratings: NaN"
```

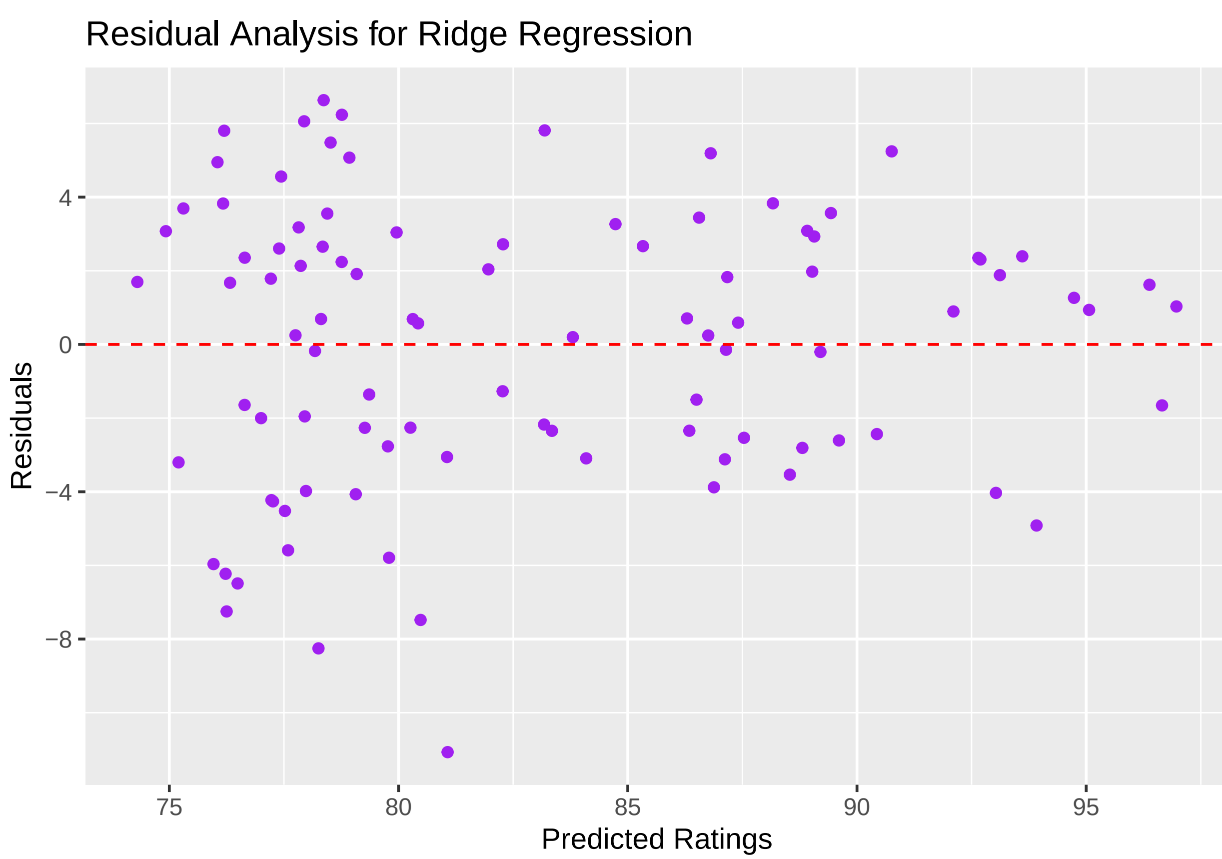
```r
print(paste("RMSE for ridge regression ratings:", rmse_ridge))
```

```
## [1] "RMSE for ridge regression ratings: NaN"
```

```r
# Fit lasso regression model
lasso_model <- cv.glmnet(x, y, alpha = 1)
players$predicted_rating_lasso <- predict(lasso_model, s = "lambda.min", newx = x)

# Compare correlations of ridge and lasso models
correlation_coefficient_lasso <- cor(players$predicted_rating_lasso, players$rating_2k)
print(paste("The correlation coefficient between the lasso regression ratings and the actual ratings is
```

```
## [1] "The correlation coefficient between the lasso regression ratings and the actual ratings is 1"
```

```r
# Calculate residuals for ridge regression
players$residuals_ridge <- players$nba_2k25_rating - players$predicted_rating_ridge

# Visualize residuals
ggplot(players, aes(x = predicted_rating_ridge, y = residuals_ridge)) +
  geom_point(color = 'purple') +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = 'Residual Analysis for Ridge Regression', x = 'Predicted Ratings', y = 'Residuals')
```




```r
# Rank players by specific skill areas
ranked_by_scoring <- players[order(-players$scoring), c("name", "scoring")]
ranked_by_defense <- players[order(-players$defense), c("name", "defense")]

print("Top Players by Scoring:")
```

```
## [1] "Top Players by Scoring:"
```

```r
print(head(ranked_by_scoring, 5))
```

```
##                        name  scoring
## 3               Joel Embiid 99.00000
## 2               Luka Doncic 98.17949
## 4     Giannis Antetokounmpo 94.58974
## 5  Shai Gilgeous-Alexander 94.28205
## 28           Damian Lillard 94.17949
```

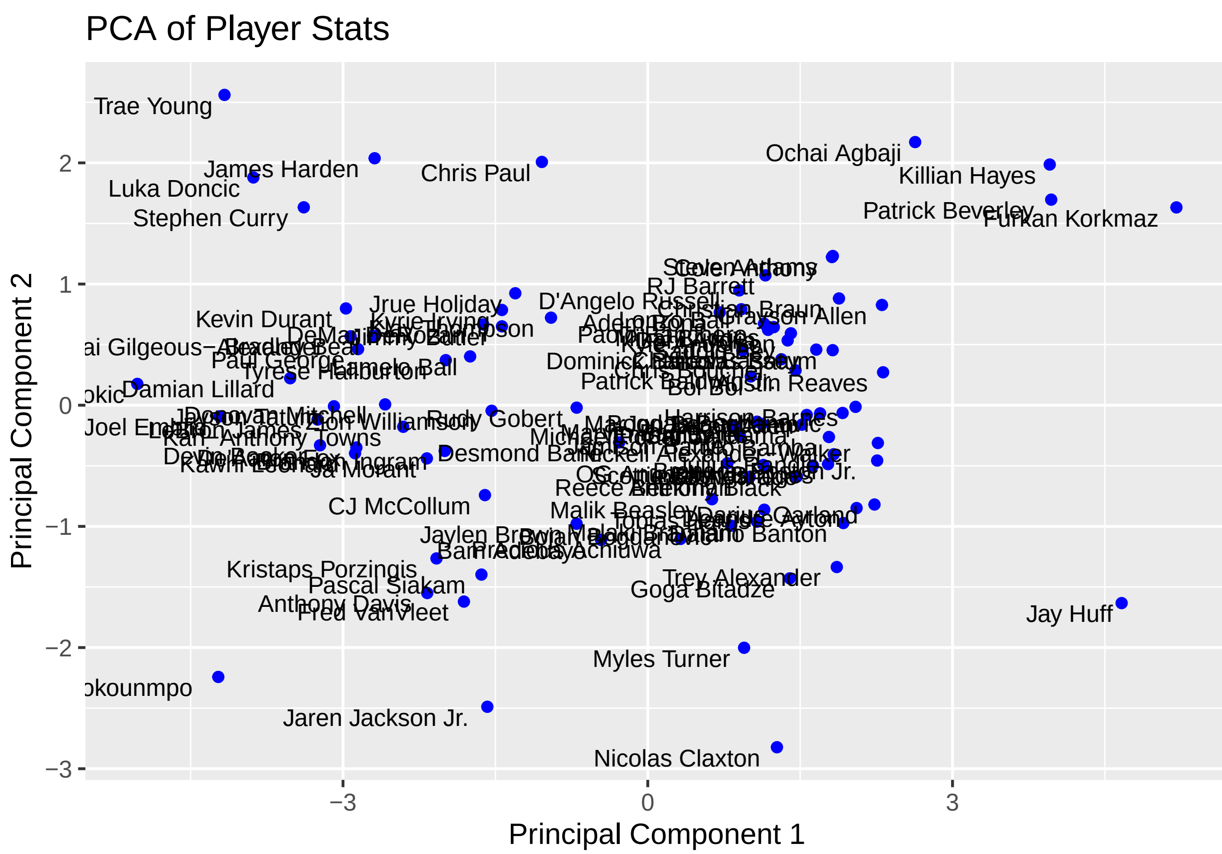
```r
print("Top Players by Defense:")
```

```
## [1] "Top Players by Defense:"
```

```r
print(head(ranked_by_defense, 5))
```

```
##                      name  defense
## 14        Nicolas Claxton 99.00000
## 4   Giannis Antetokounmpo 96.09091
## 34       Jaren Jackson Jr. 96.09091
## 13               Jay Huff 93.18182
## 18           Myles Turner 93.18182
```

```r
library(stats)

# Perform PCA
player_stats <- players[, c("points_per_game", "defensive_rating", "assists_per_game", "per", "win_shar
pca_model <- prcomp(player_stats, scale. = TRUE)

# Add PCA components to the dataset
players$PC1 <- pca_model$x[, 1]
players$PC2 <- pca_model$x[, 2]

# Visualize PCA
ggplot(players, aes(x = PC1, y = PC2, label = name)) +
  geom_point(color = 'blue') +
  geom_text(size = 3, hjust = 1.1, vjust = 1.1) +
  labs(title = "PCA of Player Stats", x = "Principal Component 1", y = "Principal Component 2")
```




```
# Fit lasso regression for comparison
lasso_model <- cv.glmnet(x, y, alpha = 1)
players$predicted_rating_lasso <- predict(lasso_model, s = "lambda.min", newx = x)

# Compare correlations
correlation_lasso <- cor(players$predicted_rating_lasso, players$rating_2k)
print(paste("The correlation coefficient for Lasso regression is", correlation_lasso))
```

```
## [1] "The correlation coefficient for Lasso regression is 1"
```

```
# Save correlation coefficients
correlation_results <- data.frame(
  Model = c("Formula", "Ridge Regression", "Lasso Regression"),
  Correlation = c(correlation_coefficient_formula, correlation_coefficient_ridge, correlation_lasso)
)
write.csv(correlation_results, "correlation_results.csv")

# Save feature importance
write.csv(correlation_coefficient_ridge, "ridge_feature_importance.csv")

# Load necessary libraries for diagnostics
library(car)  # for vif
```

```
## Warning: package 'car' was built under R version 4.3.2
```

```
## Loading required package: carData

## Warning: package 'carData' was built under R version 4.3.2
```

```r
library(MASS) # for influence measures
library(glmnet)

# Calculate Variance Inflation Factor (VIF)
vif_results <- vif(lm(y ~ ., data = as.data.frame(x)))
print("Variance Inflation Factors (VIF):")
```

```
## [1] "Variance Inflation Factors (VIF):"
```

```r
print(vif_results)
```

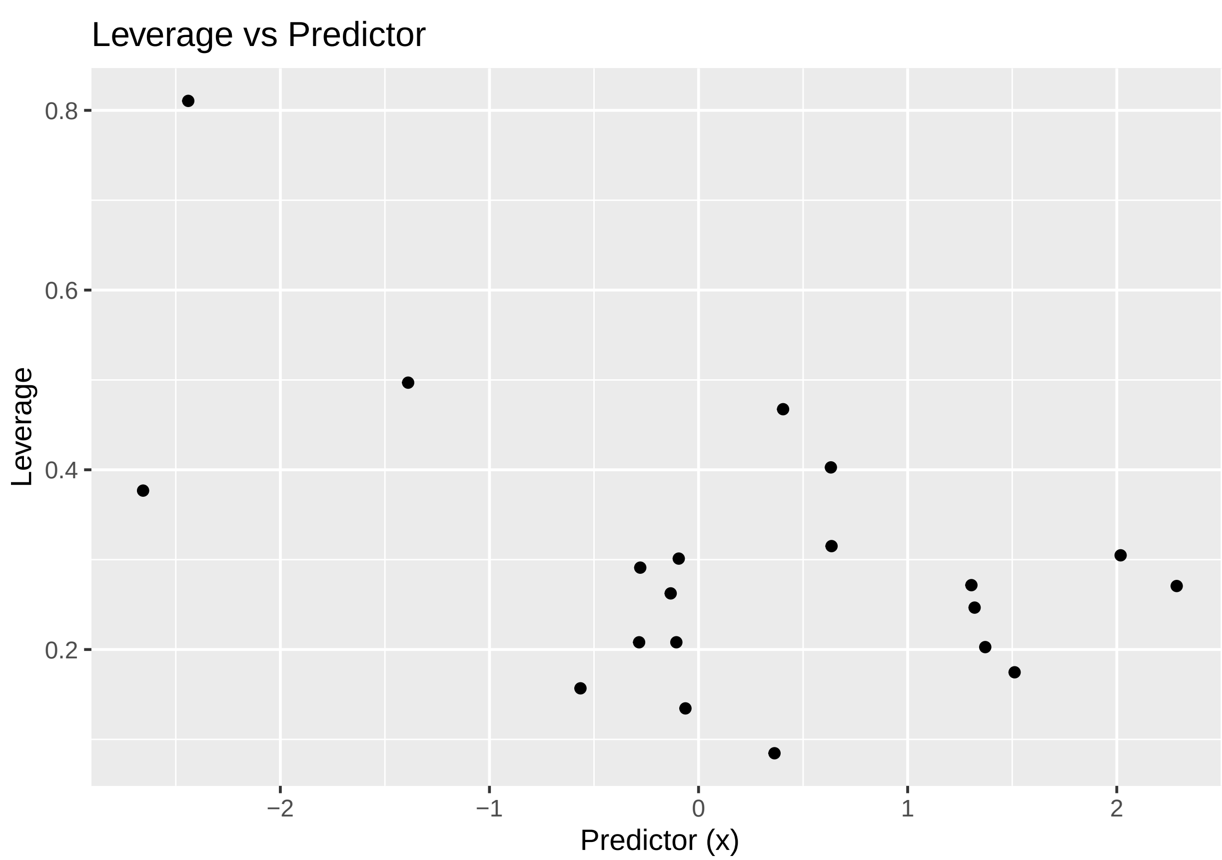
```
##   points_per_game  defensive_rating  assists_per_game               per
##          5.406883          1.143215          2.548631         14.758970
## win_shares_per_48               bpm              vorp
##         15.496876         15.224906         16.184543
```

```r
# Load necessary libraries
library(glmnet)
library(Matrix)

# Sample data
set.seed(42)
X <- as.matrix(cbind(1, matrix(rnorm(100), nrow = 20)))  # Add intercept (column of 1s)
y <- rnorm(20)

# Fit the ridge regression model
ridge_model <- glmnet(X, y, alpha = 0)  # alpha = 0 for ridge regression

# Extract the coefficients for a particular lambda (e.g., lambda = 0.1)
lambda_index <- which.min(ridge_model$lambda)  # Use the best lambda by CV or choose one manually
lambda <- ridge_model$lambda[lambda_index]

# Compute the hat matrix for ridge regression
XtX <- t(X) %*% X
XtX_lambda_inv <- solve(XtX + lambda * diag(ncol(X)))  # Regularized inverse
H_ridge <- X %*% XtX_lambda_inv %*% t(X)

# Leverage values (diagonal of the hat matrix)
leverage_values <- diag(H_ridge)

# Add leverage to the data
data <- cbind(as.data.frame(X), leverage = leverage_values)

# Display the leverage values
print(data)
```

```
##    V1           V2           V3           V4            V5           V6   leverage
```

```
## 1   1  1.37095845 -0.30663859  0.20599860 -0.36723464  1.51270701 0.20267656
## 2   1 -0.56469817 -1.78130843 -0.36105730  0.18523056  0.25792144 0.15681710
## 3   1  0.36312841 -0.17191736  0.75816324  0.58182373  0.08844023 0.08456845
## 4   1  0.63286260  1.21467470 -0.72670483  1.39973683 -0.12089654 0.40265714
## 5   1  0.40426832  1.89519346 -1.36828104 -0.72729206 -1.19432890 0.46742619
## 6   1 -0.10612452 -0.43046913  0.43281803  1.30254263  0.61199690 0.20805500
## 7   1  1.51152200 -0.25726938 -0.81139318  0.33584812 -0.21713985 0.17465298
## 8   1 -0.09465904 -1.76316309  1.44410126  1.03850610 -0.18275671 0.30116477
## 9   1  2.01842371  0.46009735 -0.43144620  0.92072857  0.93334633 0.30483533
## 10  1 -0.06271410 -0.63999488  0.65564788  0.72087816  0.82177311 0.13448300
## 11  1  1.30486965  0.45545012  0.32192527 -1.04311894  1.39211638 0.27163433
## 12  1  2.28664539  0.70483734 -0.78383894 -0.09018639 -0.47617392 0.27066875
## 13  1 -1.38886070  1.03510352  1.57572752  0.62351816  0.65034856 0.49698809
## 14  1 -0.27878877 -0.60892638  0.64289931 -0.95352336  1.39111046 0.29110109
## 15  1 -0.13332134  0.50495512  0.08976065 -0.54282881 -1.11078888 0.26245569
## 16  1  0.63595040 -1.71700868  0.27655075  0.58099650 -0.86079259 0.31507222
## 17  1 -0.28425292 -0.78445901  0.67928882  0.76817874 -1.13173868 0.20804481
## 18  1 -2.65645542 -0.85090759  0.08983289  0.46376759 -1.45921400 0.37683133
## 19  1 -2.44046693 -2.41420765 -2.99309008 -0.88577630  0.07998255 0.81052504
## 20  1  1.32011335  0.03612261  0.28488295 -1.09978090  0.65320434 0.24663689
```

```r
# Optional: Plot leverage values against the predictor variable (excluding intercept column)
library(ggplot2)
ggplot(data, aes(x = X[, 2], y = leverage)) +  # X[, 2] assumes the first column is the intercept
  geom_point() +
  labs(title = "Leverage vs Predictor", x = "Predictor (x)", y = "Leverage")
```

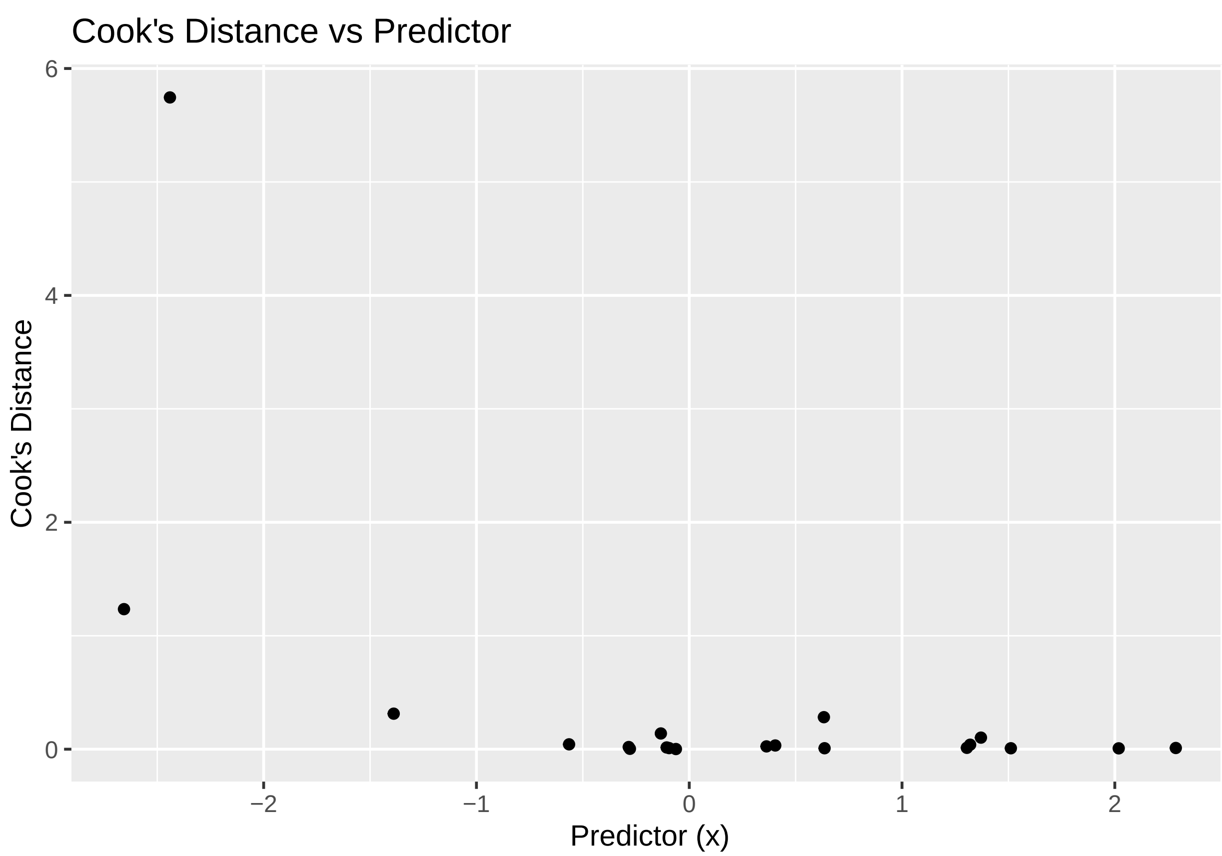


```r
# Calculate Cook's Distance
# Load necessary libraries
library(glmnet)

# Sample data
set.seed(42)
X <- as.matrix(cbind(1, matrix(rnorm(100), nrow = 20)))  # Add intercept (column of 1s)
y <- rnorm(20)

# Fit the ridge regression model
ridge_model <- glmnet(X, y, alpha = 0)  # alpha = 0 for ridge regression

# Extract the coefficients for a particular lambda (e.g., lambda = 0.1)
lambda_index <- which.min(ridge_model$lambda)  # Use the best lambda by CV or choose one manually
lambda <- ridge_model$lambda[lambda_index]

# Compute the fitted values (predictions)
fitted_values <- predict(ridge_model, s = lambda, newx = X)

# Compute the residuals
residuals <- y - fitted_values

# Compute the Mean Squared Error (MSE)
mse <- mean(residuals^2)

# Compute the hat matrix (leverage values)
XtX <- t(X) %*% X
XtX_lambda_inv <- solve(XtX + lambda * diag(ncol(X)))  # Regularized inverse
H_ridge <- X %*% XtX_lambda_inv %*% t(X)

# Leverage values (diagonal of the hat matrix)
h_ii <- diag(H_ridge)

# Compute Cook's distances
cooks_distances <- (residuals^2 / (ncol(X) * mse)) * (h_ii / (1 - h_ii)^2)

# Combine data with Cook's distances
data <- cbind(as.data.frame(X), Cook_Distance = cooks_distances)

# Display Cook's distances
print(data)
```

```
##    V1          V2          V3          V4          V5          V6          s1
## 1   1  1.37095845 -0.30663859  0.20599860 -0.36723464  1.51270701 0.102198649
## 2   1 -0.56469817 -1.78130843 -0.36105730  0.18523056  0.25792144 0.042841504
## 3   1  0.36312841 -0.17191736  0.75816324  0.58182373  0.08844023 0.024834360
## 4   1  0.63286260  1.21467470 -0.72670483  1.39973683 -0.12089654 0.282332164
## 5   1  0.40426832  1.89519346 -1.36828104 -0.72729206 -1.19432890 0.032816631
## 6   1 -0.10612452 -0.43046913  0.43281803  1.30254263  0.61199690 0.015445057
## 7   1  1.51152200 -0.25726938 -0.81139318  0.33584812 -0.21713985 0.008463473
## 8   1 -0.09465904 -1.76316309  1.44410126  1.03850610 -0.18275671 0.009788883
## 9   1  2.01842371  0.46009735 -0.43144620  0.92072857  0.93334633 0.007563423
## 10  1 -0.06271410 -0.63999488  0.65564788  0.72087816  0.82177311 0.001560409
```

```
## 11  1  1.30486965  0.45545012  0.32192527 -1.04311894  1.39211638 0.012183655
## 12  1  2.28664539  0.70483734 -0.78383894 -0.09018639 -0.47617392 0.010714406
## 13  1 -1.38886070  1.03510352  1.57572752  0.62351816  0.65034856 0.313221115
## 14  1 -0.27878877 -0.60892638  0.64289931 -0.95352336  1.39111046 0.003686553
## 15  1 -0.13332134  0.50495512  0.08976065 -0.54282881 -1.11078888 0.138648966
## 16  1  0.63595040 -1.71700868  0.27655075  0.58099650 -0.86079259 0.008575668
## 17  1 -0.28425292 -0.78445901  0.67928882  0.76817874 -1.13173868 0.019108892
## 18  1 -2.65645542 -0.85090759  0.08983289  0.46376759 -1.45921400 1.234711228
## 19  1 -2.44046693 -2.41420765 -2.99309008 -0.88577630  0.07998255 5.744897266
## 20  1  1.32011335  0.03612261  0.28488295 -1.09978090  0.65320434 0.039511083
```

```r
# Optional: Plot Cook's distance against the predictor variable (excluding intercept column)
library(ggplot2)
ggplot(data, aes(x = X[, 2], y = cooks_distances)) +  # X[, 2] assumes the first column is the intercep
  geom_point() +
  labs(title = "Cook's Distance vs Predictor", x = "Predictor (x)", y = "Cook's Distance")
```




```r
# You can filter the players dataframe to highlight high leverage and influential points
high_leverage_players <- players[players$leverage > 2 * (nrow(players) / ncol(x)), ]
high_cooks_distance_players <- players[players$cooks_distance > 1, ]
```