stat 462 group project

Brayden Adams

2024-12-08

The Code

```
# 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
```

```
library(ggplot2)

# Define player stats (including Furkan Korkmaz)
players <- data.frame(
  name = c("Nikola Jokic", "Luka Doncic", "Joel Embiid", "Giannis Antetokounmpo", "Shai Gilgeous-Alexand
           "Anthony Davis", "LeBron James", "Kevin Durant", "Jayson Tatum", "Stephen Curry", "Killian Ha
           "Patrick Beverley", "Jay Huff", "Nicolas Claxton", "Desmond Bane", "Tobias Harris", "Paolo Ba
           "Myles Turner", "Austin Reaves", "D'Angelo Russell", "Ja Morant", "Furkan Korkmaz"),
  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, 1
                      20.0, 15.0, 13.0, 17.0, 25.1, 6.8),
  defensive_rating = c(107, 110, 107, 102, 108, 104, 106, 109, 107, 111, 110, 110, 103, 101, 106, 105,
                       108, 103, 107, 108, 105, 110),
  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, 3.7, 1.2, 3.4, 6.1, 8.1, 1.2),
  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
          16.0, 18.0, 14.0, 16.5, 20.8, 11.2),
  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
                        0.100, 0.150, 0.200, 0.180, 0.150, 0.200, 0.120, 0.140, 0.180, 0.050),
  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
  rating_2k = c(98, 95, 96, 96, 93, 93, 96, 96, 95, 96, 75, 76, 67, 84, 84, 82, 84, 83, 80, 83, 92, 70)
)

# 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_shares
y <- players$rating_2k

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

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
# 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$final_rating_formula, players$rating_2k)
correlation_coefficient_ridge <- cor(players$predicted_rating_ridge, players$rating_2k)

# 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:"
```

```
print(ranked_players_formula[, c("name", "final_rating_formula", "rating_2k")])
```

```
##                        name final_rating_formula rating_2k
## 4    Giannis Antetokounmpo             93.77212        96
## 1             Nikola Jokic             92.87562        98
## 3              Joel Embiid             90.18909        96
## 2              Luka Doncic             89.22735        95
## 7             LeBron James             89.15628        96
## 9             Jayson Tatum             88.22699        95
## 5  Shai Gilgeous-Alexander             88.12878        93
## 21               Ja Morant             86.80100        92
## 6            Anthony Davis             86.69267        93
```

```
## 8             Kevin Durant             86.04824        96
## 10           Stephen Curry             85.62215        96
## 15            Desmond Bane             82.94529        84
## 14         Nicolas Claxton             81.08381        84
## 18            Myles Turner             80.67135        83
## 16           Tobias Harris             80.29408        82
## 20        D'Angelo Russell             79.47092        83
## 17          Paolo Banchero             78.44768        84
## 19           Austin Reaves             76.07607        80
## 13                Jay Huff             74.72823        67
## 11           Killian Hayes             71.17380        75
## 12        Patrick Beverley             70.07937        76
## 22          Furkan Korkmaz             68.66979        70
```

```
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 0.934285172890
```

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

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

```
print(ranked_players_ridge[, c("name", "predicted_rating_ridge", "rating_2k")])
```

```
##                       name lambda.min rating_2k
## 1             Nikola Jokic   98.79436        98
## 3              Joel Embiid   97.67555        96
## 4    Giannis Antetokounmpo   96.47991        96
## 2              Luka Doncic   96.14427        95
## 10           Stephen Curry   95.76055        96
## 7             LeBron James   94.39799        96
## 8             Kevin Durant   94.39239        96
## 9             Jayson Tatum   93.96865        95
## 5  Shai Gilgeous-Alexander   93.19572        93
## 6            Anthony Davis   92.04193        93
## 21               Ja Morant   92.00504        92
## 15            Desmond Bane   85.85183        84
## 18            Myles Turner   82.80708        83
## 20        D'Angelo Russell   82.76934        83
## 17          Paolo Banchero   82.49478        84
## 14         Nicolas Claxton   81.84000        84
## 16           Tobias Harris   81.77049        82
## 19           Austin Reaves   79.28041        80
## 12        Patrick Beverley   75.49430        76
## 11           Killian Hayes   74.81700        75
## 22          Furkan Korkmaz   71.13876        70
## 13                Jay Huff   70.87964        67
```

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

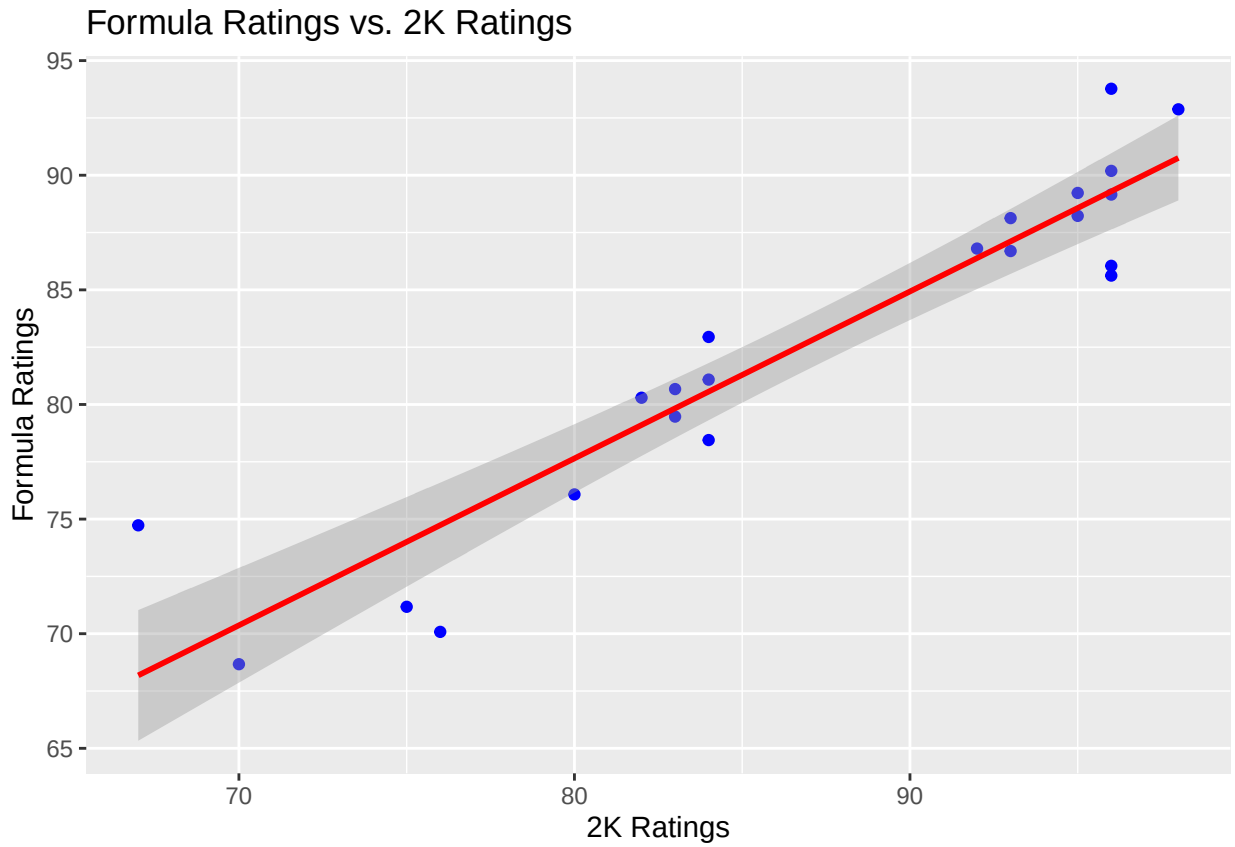
```
## [1] "The correlation coefficient between the ridge regression ratings and the actual ratings is"
```

```
correlation_coefficient_ridge
```

```
##                  [,1]
## lambda.min 0.9893865
```

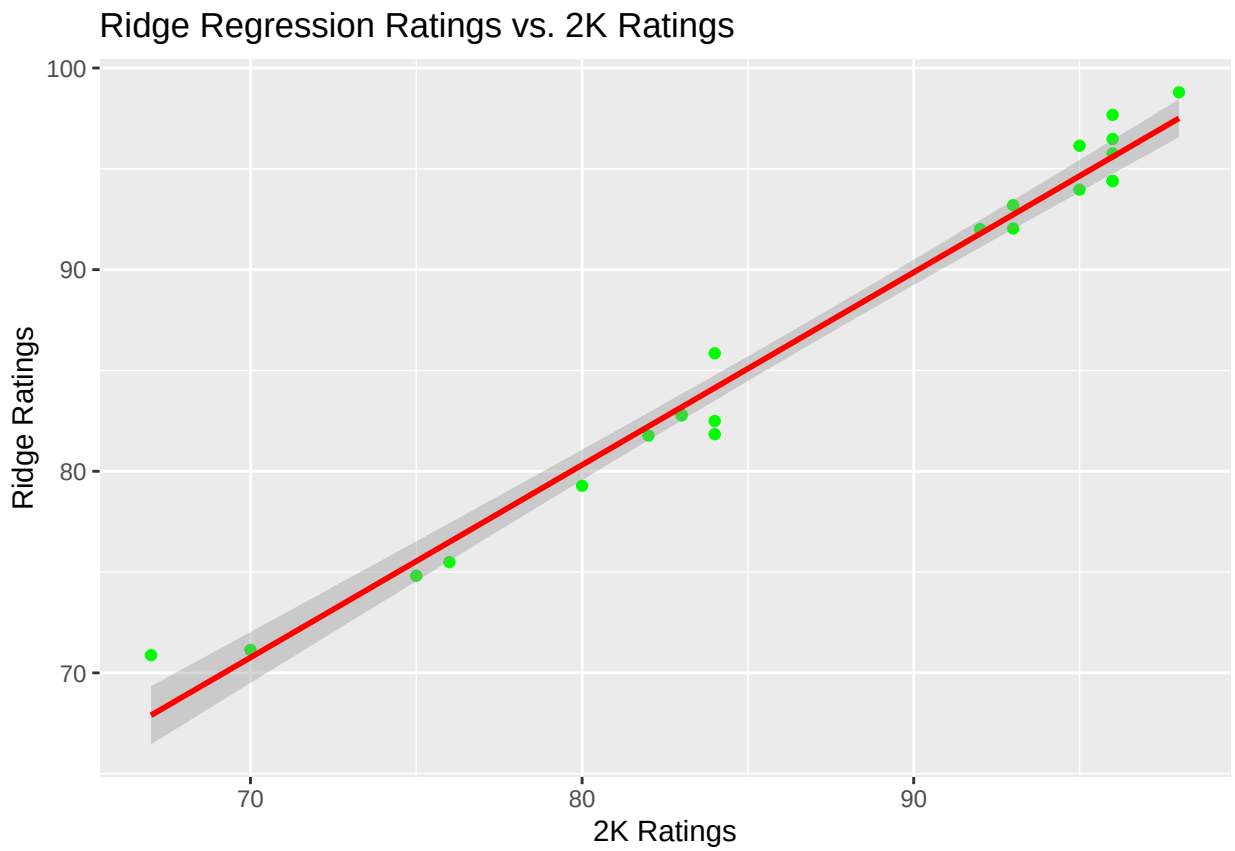
```
library(ggplot2)
ggplot(players, aes(x = rating_2k, y = final_rating_formula)) +
  geom_point(color = 'blue') +
  geom_smooth(method = 'lm', color = 'red') +
  labs(title = 'Formula Ratings vs. 2K Ratings', x = '2K Ratings', y = 'Formula Ratings')
```

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




```
ggplot(players, aes(x = rating_2k, 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'
```




```
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)       52.2352536
## points_per_game    0.1424025
## defensive_rating   0.1229892
## assists_per_game   0.1424211
## per                0.1616979
## win_shares_per_48  9.7154013
## bpm                1.2542028
## vorp               1.3689270
```

```
# 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)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
best_lambda <- ridge_model_tuned$lambda.min
print(paste("Best lambda:", best_lambda))
```

```
## [1] "Best lambda: 0.1"
```

```
# 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:"
```

```
print(coefficients_df)
```

```
##             feature importance
## 6 win_shares_per_48  9.7154013
## 8              vorp  1.3689270
## 7               bpm  1.2542028
## 5               per  0.1616979
## 4  assists_per_game  0.1424211
## 2   points_per_game  0.1424025
## 3  defensive_rating  0.1229892
```

```
# 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:"
```

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

```
##              name final_rating_formula
## 1    Nikola Jokic             92.87562
## 10  Stephen Curry             85.62215
## 22 Furkan Korkmaz             68.66979
```

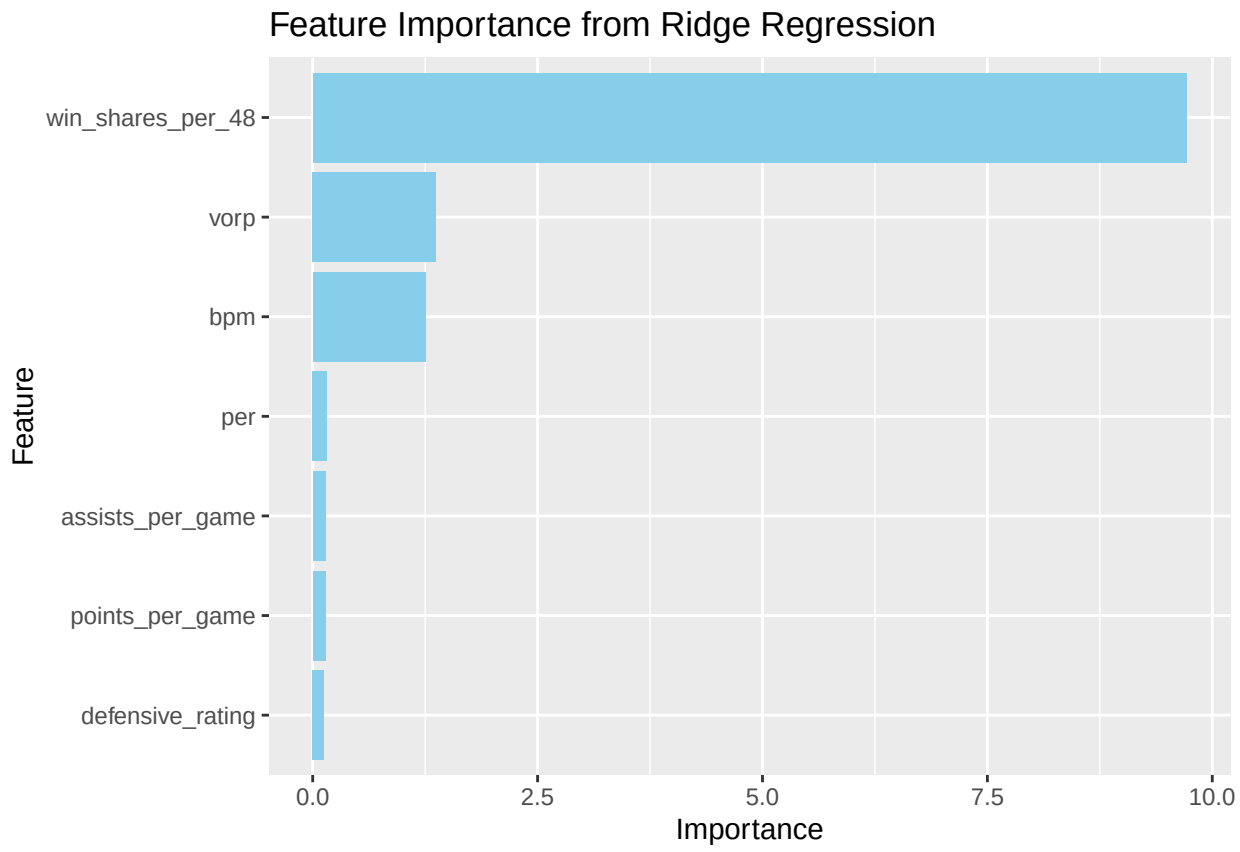
```
print("Custom Player Rankings based on Ridge Regression:")
```

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

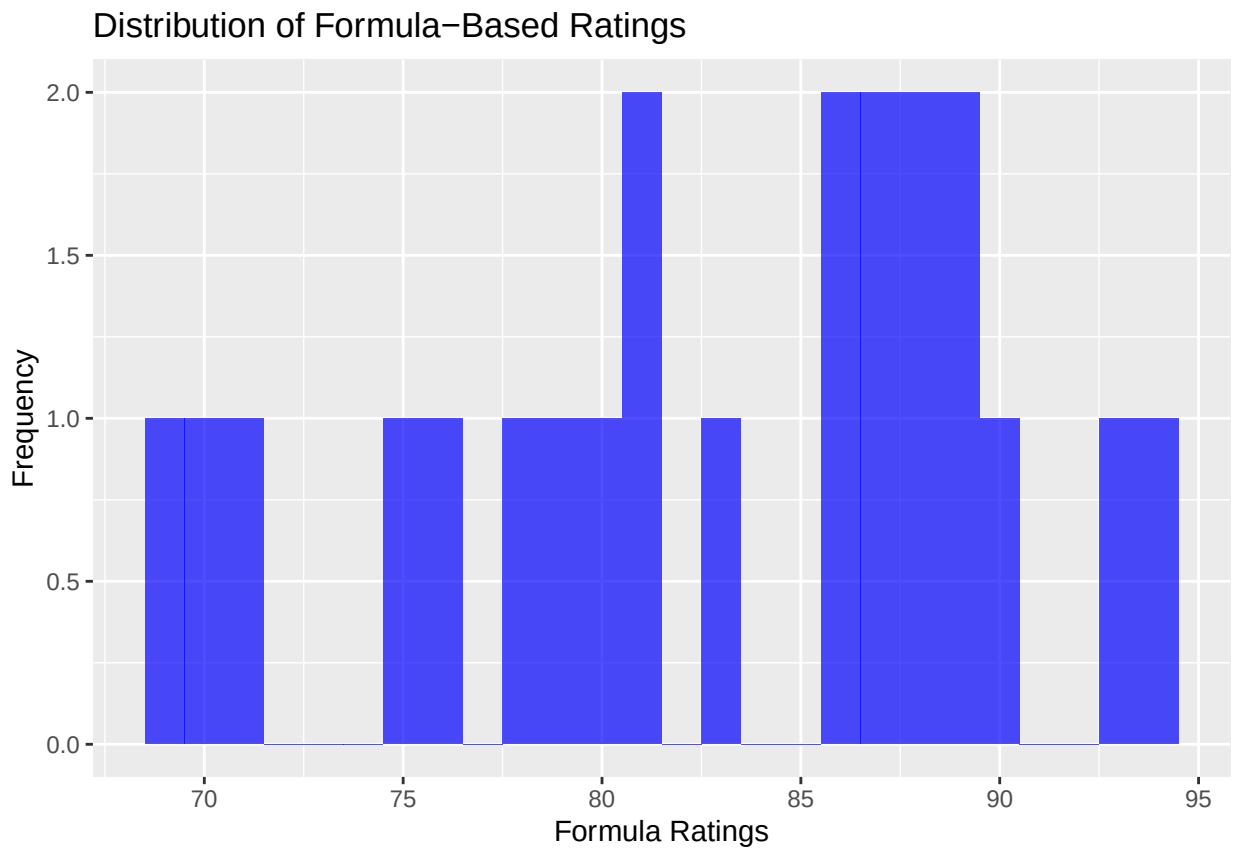
```
print(custom_players_ridge[, c("name", "predicted_rating_ridge")])
```

```
##              name lambda.min
## 1    Nikola Jokic   98.79436
## 10  Stephen Curry   95.76055
## 22 Furkan Korkmaz   71.13876
```

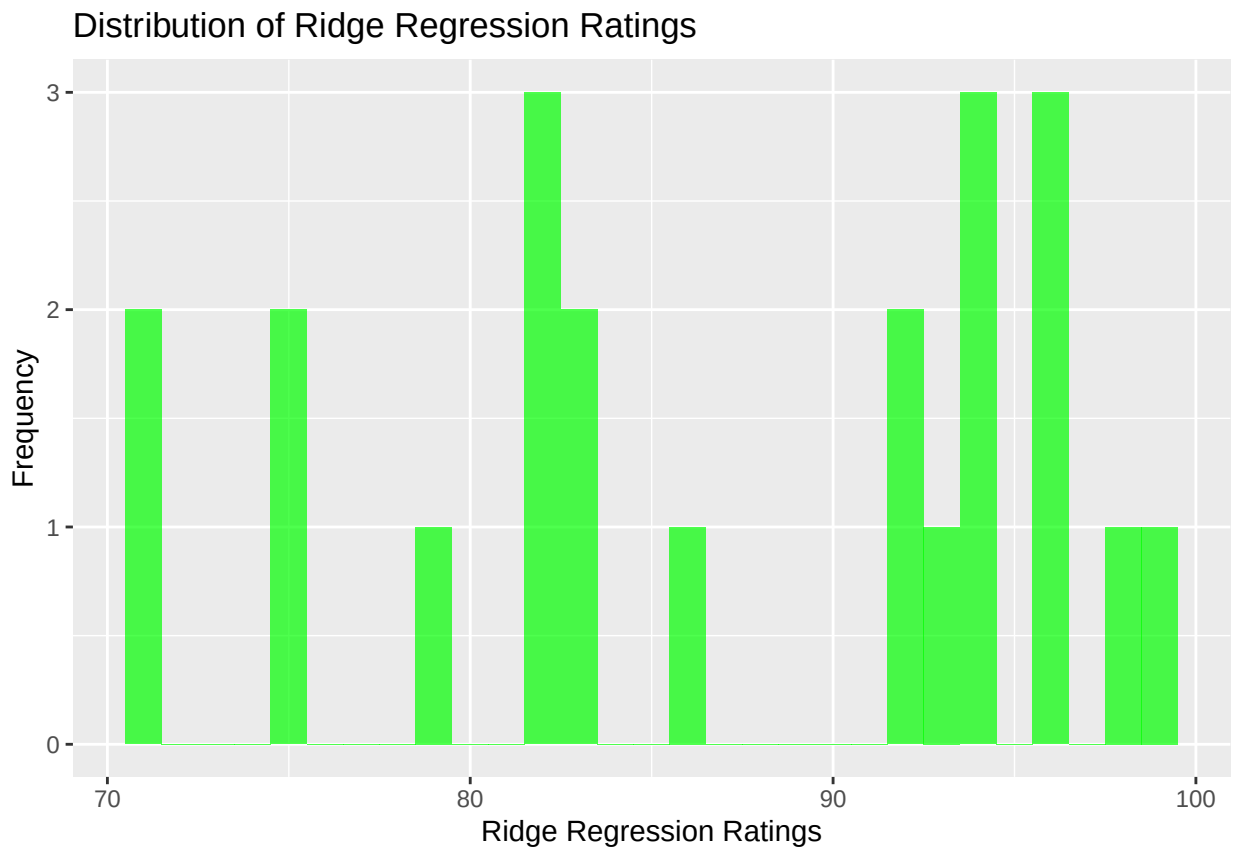
```r
# Save feature importance to CSV
write.csv(coefficients_df, "feature_importance.csv")

# 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")
```




```
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 = "Frequen
```




```
# 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: 4.95799056323504"
```

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

```
## [1] "RMSE for formula-based ratings: 5.5437074486461"
```

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

```
## [1] "MAE for ridge regression ratings: 1.01500833756959"
```

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

```
## [1] "RMSE for ridge regression ratings: 1.34612818712505"
```

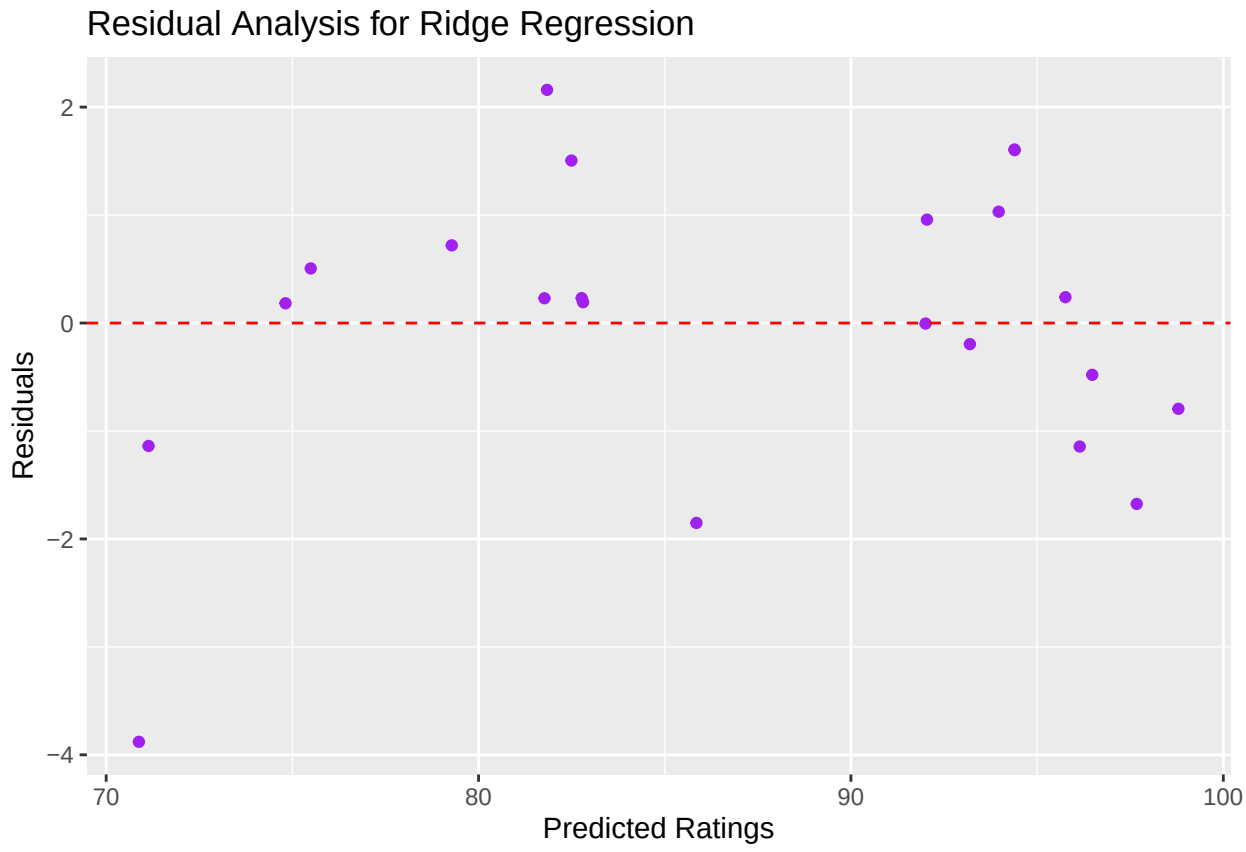
```r
# Fit lasso regression model
lasso_model <- cv.glmnet(x, y, alpha = 1)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```r
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 0.99
```

```r
# Calculate residuals for ridge regression
players$residuals_ridge <- players$rating_2k - 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
## 9             Jayson Tatum 92.53846
```

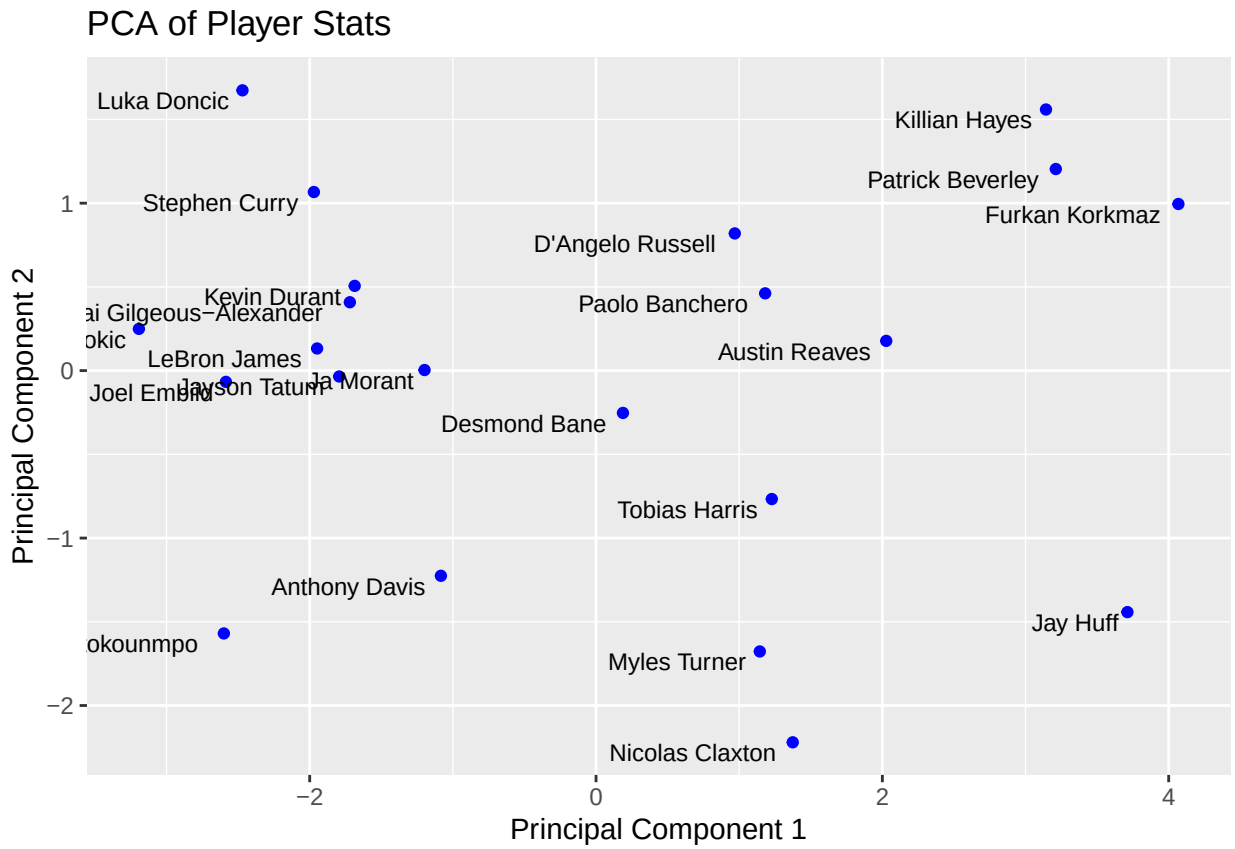
```r
print("Top Players by Defense:")
```

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

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

```
##                      name defense
## 14       Nicolas Claxton    99.0
## 4  Giannis Antetokounmpo    95.8
## 13              Jay Huff    92.6
## 18          Myles Turner    92.6
## 6           Anthony Davis    89.4
```

```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)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
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 0.992719929542386"
```

```
# 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")
```