STAT3553: Final Project - Ordinary Least Squares Analysis of Countermovement Jump Data from Carleton University Varsity Athletes

Hoang-Nam Chu, Graham Gee, Billy-Chris Muhizi

November 29, 2024

1 Introduction

1.1 Background

Among many things, Carleton University is known for its renown athletics program. Boasting 9 varsity sports, 17 club sports and 16,000 student athletes, Carleton University offers not only variety in athletic competition, but also a winning culture. Over the years, Carleton has won over 59 provincial championships and collected over 22 national titles. Perhaps greatest among all of its achievements, Carleton's Men and Women's basketball teams have amassed a large collection of championship trophies. In the last 19 years, Carleton has won 16 national championships in basketball. This is the most amount of championships won, Men or Women, for any collegiate sport in Canada or the United States of America.

Although suggesting possible causality channels can prove to be difficult, one possible reason for Carleton's historical ways could be its ability to leverage its sports data. Analyzing sports data for patterns, and then applying it is collectively known as sports analytics.

Sports analytics has a long history despite recent mainstream popularization in books and movies such as "Moneyball". Sports analytics traces its roots back to operations research performed by both American and Canadian militaries. Military statisticians performing analyses on battleground tactics lent inspiration to analysing the performance of baseball players and football teams. With the growing availability of data, tools to perform data analyses, prodigious amounts of computing power, and communication channels bypassing land and water, sports analytics, and more broadly, data analytics, has become the de facto method of drawing insights from unwieldy information.

Implementing insights from sports analytics can be seen through several layers. The largest of these is from an operational standpoint: how can data be used to increase and draw attention to the sports team. For the Carleton Ravens, this may be analysing ticket sales to Raven's games, or seeing how much merchandise is selling. One could also include scouting into this group. Data is used to identify talent nationally and internationally and attract them to our teams.

The next of these layers is increasing team performance as a unit. This includes analyzing aggregated team data for coaching staff to make adjustments in real time, and forecast well-performing line-ups for future match-ups.

Finally, the most granular level deals with individuals as the atomic unit. Sports analytics can be used to identify injury prone athletes, to rehabilitate athletes and also to improve athletes' weaknesses and reinforce their strengths.

1.2 Motivation

As head strength and conditioning coach for the entire Carleton Ravens' team rosters, Nick Westcott has to deal with a lot of moving pieces. In his own words, the role of a strength and conditioning coach is to manipulate variables tied to an athlete's strength and conditioning regime in order to enhance their short-term and long-term performance. From this standpoint, Nick's role as an affect on all three layers that we discussed above - a complex task.

The motivation behind our project started with an existing partnership between the Statistics and Athletic Departments at Carleton University. Motivated students were offered the chance to collaborate with the Athletic department to analyze data collected from Carleton's sports teams. Due to this partnership, our team was able to reach out to Nick and Shirley Mills - liaison for the Statistics Department - to inquire about performing regression analysis on one of their datasets.

1.3 The Problem

In the end, we both agreed to a particular dataset containing information about a specific strength diagnostic test called the counter-movement jump. Every varsity athlete must complete this test before their season and throughout the athletic calendar year. The importance of this test is well-documented in strength and conditioning protocol.

The countermovement jump (CMJ) is a type of strength test that measures lower body strength. Although CMJ protocols can vary, the standard protocol calls for the user to stand on two force plates with their feet shoulder-width apart. With their hands on their hips to eliminate the use of their upper body, the user descends to their preferred depth before jumping straight up as high as possible before landing on their original spot. Their hands must remain on their hips the entire duration of the jump. The CMJ is considered to be the gold standard in reliability in terms of jump tests.

Studies have shown that stronger athletes can jump higher than weaker athletes; thus the CMJ can be used as a tool for talent identification, as a tool for progression in strength, and as a tool for rehabilitation.

Although Nick had the raw data for his athlete's CMJs, we wanted to explore the relationship between an athlete's jump height and their sport. Given the time and budget constraints that Nick is under, performing strength diagnostics might be more suitable for certain sports. We sought to explore this relationship and their interactions with the metrics returned by the CMJ apparatus.

1.4 The Dataset

Nick presented us with data that he had gathered from his athletes between the years 2020-2024. The dataset contains 17290 rows of preprocessed data, with each row uniquely identifying a single jump. For each testing period, jumps were performed within 30 seconds of each other. The dataset contained over 80 metrics produced by the countermovement dual force plate system (Hawkins Dynamics LLC, Westbrook, Maine). The master data set contained data on over 635 unique athletes, with the most number of jumps performed by a single athlete at 390. 5 varsity sports were included in the dataset, including, hockey, soccer, football, rugby and basketball. Each sport had differing number of rows with football having the most with over 6000 jumps, and Men's hockey having the least with 116. Data on men and women's sports included hockey, soccer, and basketball, while football was exclusively men and rugby was exclusively women.

2 Methodology

2.1 Software Packages

The R programming language was used to perform all statistical analyses on the data set. The following R packages were used throughout the analyses:

`dplyr`: used for efficient data cleaning, `stringr`: used for manipulation of string variables, `lubridate`: used for manipulation of date-time variables, `readxl`: used to read the raw data files, `faraway`: used to run variance inflation factor tests, `reshape2`: used to for manipulating wide to long data , `lars`: used to run lasso regression, `MASS`: used for principal component analysis.

2.2 Statistical Methods

2.2.1 Linear Regression

A linear regression framework was used to investigate the problem at hand. More specifically, an Ordinary Least Squares regression model was employed. Ordinary least squares works by minimizing the sum of the least squared residuals.

The goals of a regression framework are two-fold: 1) predict values in the response variable based on a set of predictor variables, 2) explore relationships between predictor variables and the response variable.

2.2.2 High Dimension Reduction Techniques

High Dimension reduction techniques were used to eliminate multi-collinearity from the model in addition to the previously mentioned technique.

Principal Component Analysis Principal Component analysis (PCA) makes use of orthogonal vectors to select components which capture the greatest amount of variance within the Xis of our model. PCA was used to reduce the size of our original model which contained a large amount of predictors.

Scree Plots for PCA: Scree plots are an index plot of the standard deviations of the principal components, so they were used for visual examination.
χ^2 **plot for PCA:** Outliers were checked using a $\chi^2(p)$ quantile plot.

Partial Least Squares Regression: Partial Least Squares Regressions (PLSR) makes use of orthogonal vectors to select components which capture the greatest correlation between X and Y within our model. PLSR was also used to reduce the size of our original model and because of our goal of prediction.

2.2.3 Multi-collinearity

In order for the β_{OLS} estimator to have a unique solution, the columns of the design matrix must be linear independent. Thus, it is vital that predictors in the model are not perfected correlated with each other.

Multi-collinearity (MC) is the presence of one or more moderately or highly correlated predictor variables. MC causes the variance of the OLS estimator to increase.

Detection of MC:
To detect MC we calculated the variance inflation factor (VIF) that measures the inflation in variance of the estimated regression coefficient by the presence of highly correlated predictors.

$$VIF_k = \frac{Var(\hat{\beta}_k)}{Var(\hat{\beta}_k)_{min}} = \frac{1}{1 - R_k^2}$$

VIF_k scores of over 4 inclusively indicate the potential of MC while scores above 10 indicate serious MC.

Correction of MC:
MC was corrected by removing the predictors with the highest VIF scores sequentially, and then reevaluating the VIF scores of the remaining predictors.

2.2.4 Model Selection Performance Assessment

Four selection criterion were used to assess the model performance.

1. Bayesian Information Criterion (BIC). The BIC is a selection criterion that is used to evaluate the goodness of fit of our model. It penalizes model complexity to ensure that our model is not unnecessarily large. BIC was used because it penalizes larger models more heavily than AIC and is consistent for model selection when $n \to \infty$. This works well with the attributes of our data set.

$$BIC = -2logL(\hat{\theta}) + plogn \propto nlog\{RSS(\hat{\beta})/n\} + plogn$$

2. Mallow's CP (C_p)**.**
We used Mallow's C_p as it is an approximation of the prediction error associated with the LASSO regression.

$$C_p = \frac{RSS(\hat{\beta})}{s^2} + 2p - n.$$

3. Adjusted R^2**.**
Adjusted R^2 was used a goodness of fit measurement that would not increase RSS when more predictors were added to the model.

$$R^2_{adj} = 1 - \frac{RSS/(n-p-1)}{TSS/(n-1)} = 1 - \frac{MSE}{MST}.$$

2.2.5 Model Selection

Two model selection techniques were used to select our models: stepwise forward selection, and LASSO regression.

LASSO. The Least absolute shrinkage and selection operator (LASSO) estimator shrinks values of $\hat{\beta}_k$ to 0, eliminating them from the model. The estimator is obtained by minimizing

$$(Y - X\beta)^T(Y - X\beta) \text{ subject to} \sum_{j=1}^{n} |\beta_j| \leq t.$$

This method was selected was favoured over subset selection due to its ease of running in R.

2.2.6 Model Diagnostics

In addition to assessing model performance, 5 conditions were iteratively checked to ensure that the model did not violate any OLS assumptions or bias its estimate.

1. Linearity between response and predictor variables. Linearity between the response and predictor variables is the underlying basis of OLS. In order to verify this assumption, we plotted the residuals vs the fitted values.

Partial Residual plots are used to assess the linearity of the model. When plotting with termplot(), we look to see that the line created among the points follows a linear trend, as well as identifying different relationships among groups in the data. This is important as we assess the relationship of our over 80 predictors, to ensure that linearity is satisfied.

2. Homoscedasticity. Homoscedasticity is the constant variance of the model's residuals. Under constant variance of residuals, among other assumptions, the Gauss-Markov theorem holds and ensures that the estimator produced is the best unbiased linear estimator.

To verify homescedasticity, a residuals vs fitted plot was produced. This plot was used to assess homoscedasticity because the error terms are orthogonal to the fitted values under regression model assumptions. Thus, the plot should look random.

3. Normality of Error Terms. Normality is essential for hypothesis testing and confidence interval which we use for statistical regression, hence making sure this assumption holds is trivial for the analysis. The following methods are used to check for normality

1. QQ plot: if qq plot for residuals displays a straight line, the normality assumption is met otherwise, it is not

2. Shapiro-wilk test: the null hypothesis for this test is that the data is not normally distributed, hence a small p value rejects the null hypothesis confirming the normality for the data.

4. Outliers and Influential Points. The leverage hi is a measure of self-influence of yi on yi, outliers and influential points tends to have high leverage. The presence of such points can distort the distribution of data hence certain assumptions for statistical regression.

We can check for such points using:

Hatvalues() function in R to check for high leverage point with a rough rule that any point with leverage higher than 2p/n should be considered a high leverage point.

The halfnorm plot of leverages also gives graphical representation of observations with high leverages.

Rstudent() function in R gives studentized residuals of observations and an observation with studentized residual higher than 3 is considered an outlier.

Cook's distance() measures how big of an impact the removal of a point has on the data hence a good way to measure an influential point with any observation with cook's distance higher than 0.5 is to be considered an influential point.

5. Autocorrelation. Autocorrelation is the degree to which the error of a point is correlated with the error of a previous point in the data among points in the model. Problems with autocorrelation are frequently found in time series data, where the results may be dependent on prior data.

To assess autocorrelation, the Durbin-Watson test statistic can be used to assess the degree of autocorrelation in a model, if the null hypothesis is rejected, there is evidence to suggest that there is autocorrelation in the model.

2.2.7 Hypothesis Testing

Analysis of Variance (ANOVA) F-test is performed by conducting a hypothesis test using the F-statistic, rejecting the null hypothesis says that there is evidence to suggest that predictor is significant. This is important as we look to assess the significance of interactions between the many predictors in our model.

3 Results and Discussion

We started by cleaning the data first. The data cleaning process started with importing datasets for various sports and biological sex, each containing measurements related to countermovement jumps (CMJ). For each dataset, rows were filtered to retain only observations where the Type column indicated "Countermovement Jump" and where the Tags column was empty, effectively excluding irrelevant or mislabeled entries (e.g., CMJ arms). New columns, sex and sport, were added to indicate the gender and sport associated with each dataset, and these columns were relocated before the System Weight column for better organization. Additionally, all columns containing numerical data were explicitly converted to numeric types, ensuring consistency, and handling any non-numeric values (such as "N/A") introduced during data import.

After cleaning individual datasets, they were combined into a single large dataframe. From this combined dataset, irrelevant columns, such as Jump ID, were removed to create a smaller, more focused dataset containing only the sex, sport, and relevant performance metrics. The columns were then arranged alphabetically to align with the data dictionary, with numerical variables grouped after categorical ones (sex and sport). Any rows containing missing values (NA) were removed to ensure data completeness. We were able to do this because we deemed the missing data to be missing completely at random (MCAR). This step resulted in a clean, well-structured dataset, ready for analysis.

We implemented the model selection technique, LASSO, on the entire cleaned dataset. This technique returned to us the entire model with limited elimination of coefficients. Since we wanted our final model to be nimble, that is, as less complex as possible, we decided to forgo this model and remove unnecessary predictors in another fashion.

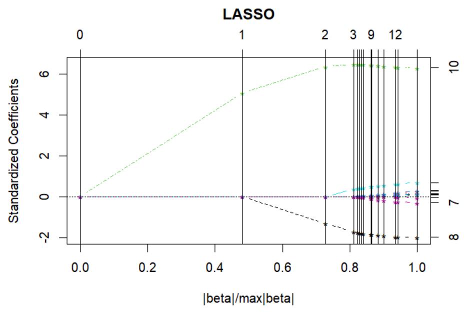
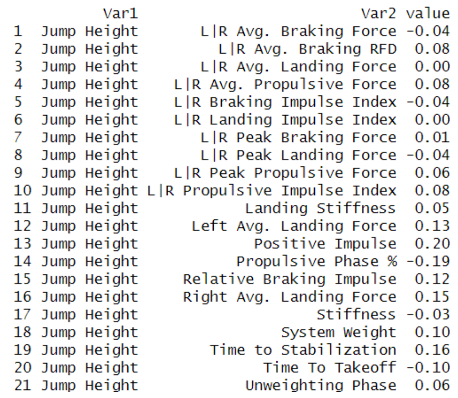


Figure 1: This is the original LASSO plot on the entire data set.

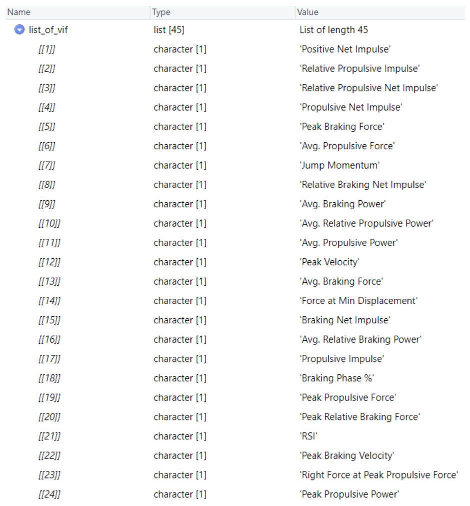
We then looked at the correlation between predictors and our response variable. To assess relationships between predictors and the response variable (Jump Height), the corr function was used to calculate a correlation matrix. This matrix was converted into a data frame to focus on the correlation between Jump Height and other predictors. Predictors with weak linear relationships (correlation values below 0.2) were removed, resulting in the elimination of 21 predictors. This reduced the number of predictors from the initial 81 to 60.



```
           Var1                          Var2 value
1   Jump Height       L|R Avg. Braking Force -0.04
2   Jump Height         L|R Avg. Braking RFD  0.08
3   Jump Height       L|R Avg. Landing Force  0.00
4   Jump Height    L|R Avg. Propulsive Force  0.08
5   Jump Height    L|R Braking Impulse Index -0.04
6   Jump Height    L|R Landing Impulse Index  0.00
7   Jump Height       L|R Peak Braking Force  0.01
8   Jump Height       L|R Peak Landing Force -0.04
9   Jump Height    L|R Peak Propulsive Force  0.06
10  Jump Height L|R Propulsive Impulse Index  0.08
11  Jump Height            Landing Stiffness  0.05
12  Jump Height      Left Avg. Landing Force  0.13
13  Jump Height             Positive Impulse  0.20
14  Jump Height           Propulsive Phase % -0.19
15  Jump Height     Relative Braking Impulse  0.12
16  Jump Height     Right Avg. Landing Force  0.15
17  Jump Height                    Stiffness -0.03
18  Jump Height                System Weight  0.10
19  Jump Height        Time to Stabilization  0.16
20  Jump Height              Time To Takeoff -0.10
21  Jump Height             Unweighting Phase  0.06
```

Figure 2: This is a correlation matrix indicating all lower correlations between our response and all other predictors.

Next, we ran parallel checks of multi-collinearity. The first of these parallel checks was by using the vif function in R, which computes the Variance Inflation Factor for each predictor. Initially, the vif function failed with the error message "non-identifiable parameters." This issue arose due to perfect collinearity among some predictors. The alias function was used to identify six predictors with perfect collinearity, which were subsequently removed. After this, vif was used iteratively, removing one predictor at a time with a VIF greater than 4. This process reduced the number of predictors further from 60 to 8. Notably, categorical predictors (sex and sport) were excluded from the VIF analysis as they were intended for categorical analysis later.



Name	Type	Value
list_of_vif	list [45]	List of length 45
[[1]]	character [1]	'Positive Net Impulse'
[[2]]	character [1]	'Relative Propulsive Impulse'
[[3]]	character [1]	'Relative Propulsive Net Impulse'
[[4]]	character [1]	'Propulsive Net Impulse'
[[5]]	character [1]	'Peak Braking Force'
[[6]]	character [1]	'Avg. Propulsive Force'
[[7]]	character [1]	'Jump Momentum'
[[8]]	character [1]	'Relative Braking Net Impulse'
[[9]]	character [1]	'Avg. Braking Power'
[[10]]	character [1]	'Avg. Relative Propulsive Power'
[[11]]	character [1]	'Avg. Propulsive Power'
[[12]]	character [1]	'Peak Velocity'
[[13]]	character [1]	'Avg. Braking Force'
[[14]]	character [1]	'Force at Min Displacement'
[[15]]	character [1]	'Braking Net Impulse'
[[16]]	character [1]	'Avg. Relative Braking Power'
[[17]]	character [1]	'Propulsive Impulse'
[[18]]	character [1]	'Braking Phase %'
[[19]]	character [1]	'Peak Propulsive Force'
[[20]]	character [1]	'Peak Relative Braking Force'
[[21]]	character [1]	'RSI'
[[22]]	character [1]	'Peak Braking Velocity'
[[23]]	character [1]	'Right Force at Peak Propulsive Force'
[[24]]	character [1]	'Peak Propulsive Power'

Figure 3: These are all the predictors that were dropped due to high VIF values.

Another test for multi-collinearity was the Principal Components Analysis Algorithm (PCA) check. With this analysis, 95% of the variations of the predictors were explained by the first 5 principal components. The associated rotation matrix was extracted from these 5 components which yielded 10 predictors.

Finally, the last multi-collinearity test was the Partial Least Squares Regression (PLSR) test which yielded another ten predictors. Once again, 90% of the correlation between the response and our predictors were explained by tenth component.

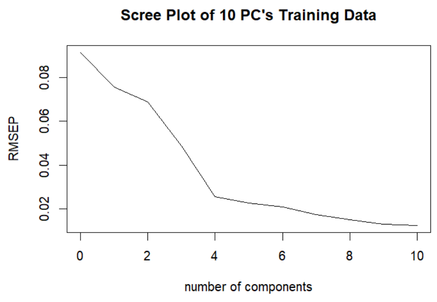


Figure 4: This is the scree plot that shows that most of the correlation was captured by the 10th component.

A 9-fold cross-validation was performed to verify that the model was not overfitting and this was confirmed with similar RMSE values.

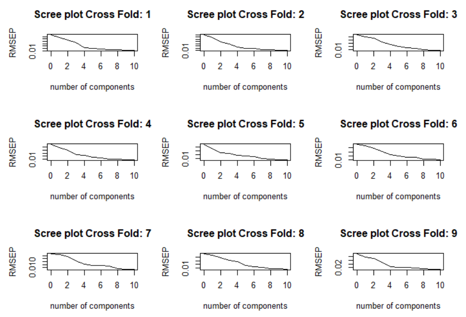
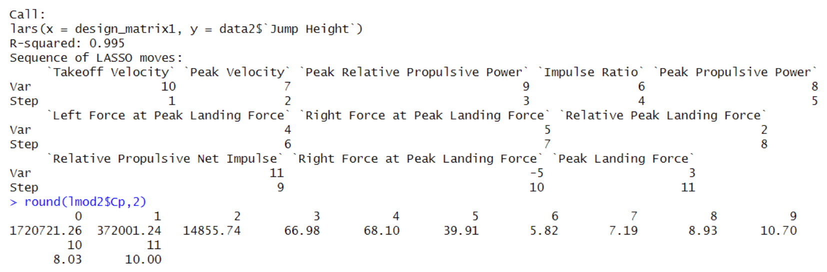


Figure 5: These screen plots show that the RMSE was roughly the same.

Given these three separate list of predictors, we presented our findings to our client, Nick, to confirm whether or not he uses any of these metrics (predictors) in his own analysis. 6 of the 10 predictors remaining from the PCA test were highlighted as being used by Nick. As opposed to 2 and 0 for the vif test and plsr tests, respectively.

With this in mind, we narrowed our selection to both the PCA and the VIF approaches. We ran another LASSO regression on the predictors remaining from both the PCA and VIF approaches. The best models according to approach and verified by LASSO was that the 6th predictor entering the model was the best for PCA and the full model was the best for VIF.



```
Call:
lars(x = design_matrix1, y = data2$`Jump Height`)
R-squared: 0.995
Sequence of LASSO moves:
     `Takeoff Velocity` `Peak Velocity` `Peak Relative Propulsive Power` `Impulse Ratio` `Peak Propulsive Power`
Var                  10               7                                9               6                        8
Step                  1               2                                3               4                        5
     `Left Force at Peak Landing Force` `Right Force at Peak Landing Force` `Relative Peak Landing Force`
Var                                   4                                   5                             2
Step                                  6                                   7                             8
     `Relative Propulsive Net Impulse` `Right Force at Peak Landing Force` `Peak Landing Force`
Var                                 11                                  -5                    3
Step                                 9                                  10                   11
> round(lmod2$Cp,2)
         0          1          2          3          4          5          6          7          8          9
1720721.26  372001.24   14855.74      66.98      68.10      39.91       5.82       7.19       8.93      10.70
        10         11
      8.03      10.00
```

Figure 6: This shows that the best model for the PCA approach was when the sixth predictor entered the model.

After running the LASSO, we ran model diagnostics. As a reminder, the diagnostics were:

1. Linearity
2. Constance Variance (Homoscedasticity)
3. Normality
4. Autocorrelation
5. Outliers / Influential Points

Diagnostic plots were generated to assess the different model assumptions for the 2 models, the vif model and the pca model.

The following were the results obtained on the vif model: Constant Variance: The residual plot was examined to check for randomness, a hallmark of constant variance. the residuals exhibited a non-random pattern, indicating that the constant variance assumption was violated.

Normality: A Q-Q plot was generated to check if residuals followed a normal distribution. The Q-Q plot deviated significantly from a straight line, confirming that the normality assumption was not satisfied. A Shapiro-Wilk test could not be performed due to the dataset exceeding the test's maximum size limit of 5000.

Influential Points:the Residuals vs. Leverage plot indicated the presence of influential points, as some extreme points were apparent.

b) on the PCA model:
Constant variance: the residual vs fitted plot exhibited a non random pattern, hence the homoscedasticity assumption not satisfied.

Normality: A Q-Q plot showed a serious deviation from the straight line, hence the normality assumption did not hold as well

influential points: the residual vs leverage plot shows that there is a presence of influential points.

c) on the PLSR model:
Similarily to the pca model, the plsr model exhibited a non random pattern on the residual vs fitted plot, and did not provide us with enough predictors that were adequately meaningful. The predictors only covered 1-2 of the 5 sections of the CMJ test, and hence this model was discarded.

Focusing on the model obtained through vif since the model obtained through piece component analysis (pca) model did not meet any model assumption. These findings prompted further refinement of the dataset and investigation into outliers and influential points.

A custom function was written to remove outliers by checking the studentized residuals of the model using the rstudent() function in R. Residuals with an absolute value of 3 or greater were identified as outliers. The function was applied within a while loop that iteratively refined the dataset. The loop terminated when no studentized residuals fell outside the range of [-3, 3]. This process removed 181 outliers from the dataset.

The final model (mod4) was then fit to the cleaned data, and diagnostic plots were generated. These plots showed an improvement in meeting model assumptions such as linearity, normality, and homoscedasticity.

To address influential points, 15 specific points were identified from the Residuals vs. Leverage plot as having a high impact on the model, as they stand out as influencer points (These points were reducing the maximum Cook's distance significantly). These points were manually removed from the dataset to enhance model stability. A new model (mod4) was fitted to this refined dataset, and Cook's distance was recalculated. A half-normal plot of Cook's distances confirmed that the remaining points were reasonable, suggesting the influence of extreme observations was successfully minimized.

The Box-Cox transformation was then applied to the response variable (Jump Height) to improve normality. Using the boxcox function, the optimal transformation parameter (lambda) was determined to be 0.9. A transformed response variable, Jump HeightBX, (The explanatory variables remain the same, excluding sex and sport) was created using this parameter, and a new model (mod5) was fit.

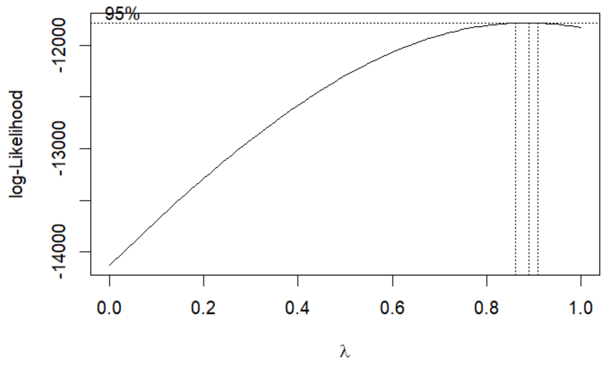


Figure 7: This is our box-cox plot that shows that the optimal lambda is around 0.9.

Although the R-squared value increased significantly to 0.9995, indicating a nearly perfect fit, however diagnostic plots revealed issues with residual patterns and other assumptions, suggesting overfitting. Consequently, the Box-Cox transformation was deemed inappropriate.

A square root transformation was subsequently applied to the response variable, creating a new model (mod6). Diagnostic plots showed similar issues as the Box-Cox transformation: improved R-squared values but residual patterns indicating overfitting; The model fits the data too closely, likely distorting generalizability. Model diagnostics (e.g., residual patterns or assumption checks) indicate poor performance despite a better R-squared value. As a result, the square root transformation was also rejected.

The process included systematic variable selection, outlier removal, and the evaluation of model assumptions. Despite attempts to improve model fit through transformations (Box-Cox and square root), both were found to overfit the data. The refined model (mod4), with 10 predictors and adjustments for outliers and influential points, provides a reasonable baseline for further analysis without introducing overfitting.

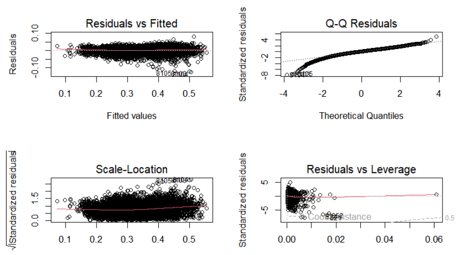


Figure 8: These were our diagnostic results before removal of outliers.

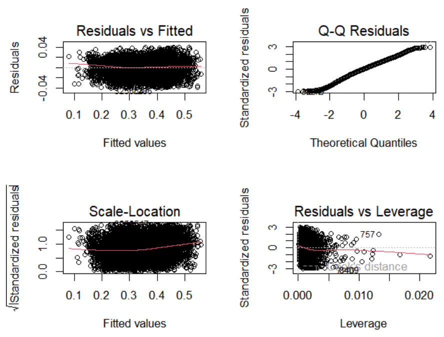


Figure 9: These are the diagnostic results after removing outliers.

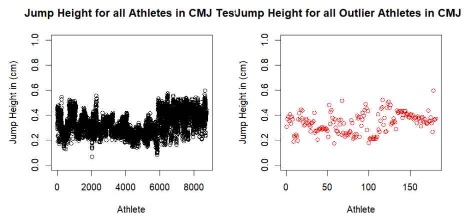


Figure 10: These are all the outliers that we removed from our model.

4 Conclusion

After testing if there is a linear relationship between jump height and Several continuous predictors provided, our analysis narrowed down these predictors to just seven of them which:are Average Landing Force, Countermovement Depth, and Peak Relative Propulsive Power, impulse ratio, relative peak landing force, right average braking RFD and unweighting phase percentage, these variables contribute significantly to explaining "Jump Height."

This will help our client save time predicting jump height for athletes, which is crucial as one of the universal performance indicators across many sports; since instead of recording 87 different data, he could emphasize the variables obtained through the model to estimate jump height.

In addition to this, considering interaction with sport as categorical data, we observe that the expected jump height of the average player varies across different sports at Carleton University.

5 References

Hawkin Dynamics Blog on Countermovement Jump Metrics: Hawkin Dynamics. (2019). Countermovement jump metrics explained. Retrieved from https://www.hawkindynamics.com/blog/co
jump-metrics-2019

Carleton University Ravens Athletics Website: Carleton University. Go Ravens. Retrieved from https://goravens.ca/

Smithsonian Lemelson Center on Sports Analytics: Smithsonian Institution. Sports analytics: Moneyball and beyond. Retrieved from https://invention.si.edu/invention-stories/sports-analytics-moneyball

Appendix

```r
# STAT 3553 Group Project on CMJ Jump Authors: Graham Gee,
# Hoang-Nam Chu, Billy Chris Muhizi

library(tidyverse)
library(readxl)
library(lars)
library(faraway)
library(MASS)
library(pls)
library(lmtest)
library(Rcmdr)

set.seed(1234)
# Loading in the data
# ------------------------------------------------

m_basket <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Men's Basketball_CMJ.xls
w_basket <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Women's Basketball_CMJ.
m_soc <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Men's Soccer_CMJ.xlsx")
w_soc <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Women's Soccer_CMJ.xlsx")
m_hock <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Men's Hockey_CMJ.xlsx")
w_hock <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Women's Hockey_CMJ.xlsx")
m_foot <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Football_CMJ.xlsx")
w_rug <- read_excel(path = "~/Data Analytics/STAT 3553/Project/CMJ_IMTP Data/Women's Rugby_CMJ.xlsx")

# Cleaning data
# -----------------------------------------------------------

# Here I've used the maggritr pipe (|> or %>%) operator
# from the dyplr package. This allows me to efficiently
# declare commands. It can be read as 'and then'. I've also
# used a flurry of dplyr subsetting functions that allow me
# to perform data cleaning operations.

# The first cleaning operation can be read the following:

# Initialize a dataframe called m_basket and then read in
# data from m_basket `and then (|>)` keep any rows that
# have the Type column equal to 'Countermovement Jump' and
# have NA values in the Tags column - essentially get rid
# of any rows that have CMJ arms or whatever - `and then`
# create two new columns called sex and sport with the
# values 'm' and 'basketball' respectively `and then` move
# those columns before system weight `and then` convert all
# columns after system weight to be numeric. The last step
# is necessary because 'N/A's in the excel form are
# registered as characters in the R.

m_basket <- m_basket |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "m", sport = "basketball") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))  #across lets me perform the mutate function across multiple colu

w_basket <- w_basket |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "w", sport = "basketball") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

m_soc <- m_soc |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "m", sport = "soccer") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

w_soc <- w_soc |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "w", sport = "soccer") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

m_hock <- m_hock |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "m", sport = "hockey") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

w_hock <- w_hock |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "w", sport = "hockey") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

w_rug <- w_rug |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "w", sport = "rugby") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

m_foot <- m_foot |>
    filter((Type == "Countermovement Jump") & is.na(Tags)) |>
    mutate(sex = "m", sport = "football") |>
    relocate(sex, sport, .before = `System Weight`) |>
    mutate(across(12:90, as.numeric))

par(mfrow = c(1, 1))

# Here I am binding all the dataframes together into one
# large dataframe.
data <- bind_rows(m_basket, w_basket, m_soc, w_soc, m_hock, w_hock,
    w_rug, m_foot)

# Here I've subsetted our large data set into a smaller one
# that contains only the columns from sex on wards. Clearly
# we don't need the jump id and so forth.
data2 <- data |>
    dplyr::select(sex:`Relative Propulsive Net Impulse`)

# I'm rearranging the columns here so that they are listed
# alphabetically to better match the data dictionary
data2 <- data2 |>
    mutate(sex = as.factor(sex), sport = as.factor(sport)) |>
    dplyr::select(order(colnames(data2))) |>
    relocate(where(is.numeric), .after = where(is.factor)) |>
    na.omit()  # this last line omits rows with NA's in them (need to look at this again)

rm(data, m_basket, w_basket, m_soc, w_soc, m_hock, w_hock, w_rug,
    m_foot)

# Basic plot of all jump heights
# ------------------------------------------------
plot(data2$`Jump Height`, ylim = c(0, 1), xlab = "Athlete", ylab = "Jump Height in (cm)",
    main = "Jump Height for all Athletes in CMJ Test")

# LASSO on whole model
# ------------------------------------------------

design_matrix0 <- model.matrix(~. - 1, data = data2)
# modeDat5<-mode(design_matrix0)

indexJH <- which(colnames(design_matrix0) == "`Jump Height`")


lmod <- lars(design_matrix0[, -indexJH], data2$`Jump Height`)
lmod
round(lmod$Cp, 2)

plot(lmod)

# Multicollinearity checks
# ------------------------------------------------

library(reshape2)
cor_2 <- round(cor(data2[3:81], use = "complete.obs"), 2)

CM <- cor_2

CM[lower.tri(CM, diag = TRUE)] <- NA

threshold <- 0.2
```

```r
cor_predictors <- subset(melt(CM, na.rm = TRUE), abs(value) <=
    (threshold))

cor_predictors_2 <- cor_predictors |>
    filter(Var1 == "Jump Height")


data3 <- data2 |>
    dplyr::select(-c(cor_predictors_2$Var2))


# VIF Check
# ------------------------------------------------------------------

# mod_3 <- lm(`Jump Height` ~ sex + sport + data3$`Avg.
# Landing Force` + data3$`Avg. Braking Force` + data3$`Avg.
# Braking Power` + data3$`Avg. Braking Velocity`+
# data3$`Avg. Propulsive Force` + data3$`Avg. Propulsive
# Power` +data3$`Avg. Propulsive Velocity` + data3$`Avg.
# Relative Braking Force` + data3$`Avg. Relative Braking
# Power` + data3$`Right Avg. Braking Force` +
# data3$`Takeoff Velocity` + data3$RSI + data3$`Right Force
# at Peak Propulsive Force` + data3$`Right Force at Peak
# Landing Force` +data3$`Right Force at Peak Braking
# Force`+data3$`Right Avg. Propulsive Force`+data3$`Right
# Avg. Braking RFD`+data3$`Right Avg. Braking
# Force`+data3$`Relative Propulsive Net Impulse` +
# data3$`Relative Propulsive Impulse` +data3$`Relative Peak
# Landing Force`+data3$`Relative Force at Min
# Displacement`+data3$`Relative Braking Net
# Impulse`+data3$`Propulsive Net Impulse`+data3$`Peak
# Velocity`+data3$`Peak Relative Propulsive
# Power`+data3$`Peak Relative Propulsive Force`+data3$`Peak
# Relative Braking Power`+data3$`Peak Relative Braking
# Force`+data3$`Peak Propulsive Power`+data3$`Peak
# Propulsive Force`+data3$`Peak Landing Force`+data3$`Peak
# Braking Velocity`+data3$`Peak Braking Power`+data3$`Peak
# Braking Force`+data3$mRSI+data3$`Jump Momentum`+
# data3$`Avg. Relative Propulsive Force` + data3$`Avg.
# Relative Propulsive Power` + data3$`Braking Impulse` +
# data3$`Braking Net Impulse` + data3$`Braking Phase` +
# data3$`Braking Phase %` + data3$`Braking RFD` +
# data3$`Countermovement Depth` + data3$`Flight Time` +
# data3$`Force at Min Displacement` + data3$`Impulse
# Ratio`, data = data3)



data4 <- data3 |>
    dplyr::select(-c(`Left Force at Peak Propulsive Force`, `Left Force at Peak Braking Force`,
        `Left Avg. Propulsive Force`, `Left Avg. Braking RFD`,
        `Left Avg. Braking Force`, `Left Force at Peak Landing Force`))
```

```r
mod_1 <- lm(`Jump Height` ~ . - sex - sport, data = data4)

vif(mod_1)

list_of_vif <- list()
high_vif <- str_remove_all(names(which.max(vif(mod_1))), "`")
list_of_vif <- append(list_of_vif, high_vif)

data4 <- data4 |>
    dplyr::select(-c(high_vif))

mod_2 <- lm(`Jump Height` ~ . - sex - sport, data = data4)

repeat {
    vif(mod_2)
    high_vif <- str_remove_all(names(which.max(vif(mod_2))),
        "`")
    list_of_vif <- append(list_of_vif, high_vif)

    data4 <- data4 |>
        dplyr::select(-c(high_vif))

    mod_2 <- lm(`Jump Height` ~ . - sex - sport, data = data4)
    vif(mod_2)

    if (max(vif(mod_2)) < 4) {
        break
    }
}
vif(mod_2)

# Model Diagnostics After MC and LASSO
# ---------------------------------------------------------

mod_3 <- lm(`Jump Height` ~ . - sex - sport, data = data4)
mod_residuals <- residuals(mod_3)
mod_fitted <- fitted(mod_3)

length(mod_residuals)
length(mod_fitted)

plot <- ggplot(data4, aes(x = mod_fitted, y = mod_residuals)) +
    geom_point() + xlab("Model Fitted Values") + ylab("Model Residuals") +
    ggtitle("Fitted vs Residual Plot")
geom_hline(yintercept = 0)
plot

# Check the normality assumption
# ------------------------------------------

qqnorm(mod_residuals, ylab = "Residuals", main = "", lwd = 2)
qqline(mod_residuals)  # clearly this doesn't look normal.
```

```r
# shapiro.test(mod_residuals) # This won't work because
# Shapiro Test only takes sample size between 3 and 5000.

par(mfrow = c(2, 2))
plot(mod_3)
# We can observe that the model is linear, but normality is
# not satisfied. The Scale-Location graph we can see is not
# linear, so there is likely some homoscedasticity. The
# Residuals vs Leverage plot has some extreme points so
# some influential points present and possible outliers.
par(mfrow = c(1, 1))

# Check for linearity
# ------------------------------------------
par(mfrow = c(1, 1))
termplot(mod_3, partial.resid = TRUE, pch = 16)

# Remove Outliers
# ------------------------------------------

removeOutl <- function(dat) {
    # Function to remove outliers
    mod <- lm(`Jump Height` ~ . - sex - sport, data = dat)
    studresiduals <- rstudent(mod)
    dat <- dat[abs(studresiduals) < 3, ]
    print(range(rstudent(mod)))

    return(dat)

}
data5 <- data4  #New dataset without outliers
mod_4 <- lm(`Jump Height` ~ . - sex - sport, data = data5)

while (TRUE) {

    data5 <- removeOutl(data5)
    mod_4 <- lm(`Jump Height` ~ . - sex - sport, data = data5)
    if (abs(range(rstudent(mod_4))[1]) > 3 || abs(range(rstudent(mod_4))[2]) >
        3) {
        print("Continue loop to eliminate more outliers")

    } else {
        break
    }

}

print("Removed all outliers")
print(range(rstudent(mod_4)))

# Get the removed values
removedVals <- setdiff(data4, data5)
```

```r
par(mfrow = c(2, 2))
plot(mod_4)  #Observe new model without outliers

# Checking what values are Outliers
# ----------------------------------------------
par(mfrow = c(1, 1))
plot(data2$`Jump Height`, ylim = c(0, 1), xlab = "Athlete", ylab = "Jump Height in (cm)",
    main = "Jump Height for all Athletes in CMJ Test")
plot(removedVals$`Jump Height`, ylim = c(0, 1), col = "red",
    xlab = "Athlete", ylab = "Jump Height in (cm)", main = "Jump Height for all Outlier Athletes in CMJ

# Check for autocorrelation
# ----------------------------------------------

dwtest(mod_2)  #Many predictors
dwtest(mod_3)  #With outliers, low amount of predictors
dwtest(mod_4)  #Without outliers
# While these indicators point towards there being
# autocorrelation, there shouldn't be any significant issue
# in the data, as each jump is an individual person, and
# would not affect the jump of anyone else. It may be the
# case that there are clusters of data that are similar,
# such as sport type, will test model with interactions to
# see if it performs better.

# Check for influential points
# -------------------------------------------

data5 <- data5[c(-911, -771, -243, -758, -8415, -7301, -755,
    -839, -8084, -897, -99, -4475, -4615, -1932, -6888), ]  #Identified 15 points that reduced max cook

mod_4 <- lm(`Jump Height` ~ . - sex - sport, data = data5)  #Refit model

par(mfrow = c(1, 1))
cookDat <- cooks.distance(mod_4)  #find Cook’s distance
# half-normal plot of Cook’s distance
halfnorm(cookDat, 3, ylab = "Cook’s distances", xlim = c(0,
    4))  #Given the removed points, this should be reasonable, without comprimising results.

par(mfrow = c(2, 2))
plot(mod_4)
# Eliminating influential points did not result in any
# significant change, likely due to the size of the
# dataset.

# Box-Cox Transformation
# ---------------------------------------------------------------

par(mfrow = c(1, 1))
mod_5 <- mod_4  #New model mod_5 for box-cox transformation
boxcoxTransform <- boxcox(mod_5, plotit = TRUE, lambda = seq(0,
    1, by = 0.1))
# Is closest to 0.9 so will try lambda=0.9
# data5$`Jump HeightBX` <- (data5$`Jump Height`^0.9 - 1) /
# 0.9 mod_5<-lm(`Jump HeightBX` ~ .-sex -sport, data =
# data5) #Fitting model with Box-cox transformation.
# par(mfrow=c(2,2)) plot(mod_5)#We can conclude this
# section as a Box-Cox Transformation is overfitting the
# data. R^2 increaded to 0.9995, but now all other metrics
# are worse. mod_5 is deemed irrelevant

# Square root Transformation
# ------------------------------------------------------------------

# mod_6<-lm(sqrt(`Jump Height`) ~ .-sex -sport, data =
# data5) #Fitting model with transformation. plot(mod_6)
# #We can also conclude that this transformation is
# overfitting in a similar way to box-cox.

# LASSO
# ------------------------------------------------------------------

design_matrix <- model.matrix(~. - 1, data = data5)
# modeDat5<-mode(design_matrix)

indexJH <- which(colnames(design_matrix) == "`Jump Height`")


lmod <- lars(design_matrix[, -indexJH], data5$`Jump Height`)
lmod
round(lmod$Cp, 2)

# PCA Test
# -----------------------------------------------------------------

# Will start by using data3
data3Scaled <- scale(data3[, c(-1:-2, -23)])
pcaTestData3 <- prcomp(data3Scaled, center = TRUE, scale. = TRUE)
print(summary(pcaTestData3))

# Try with data4 after vif eliminations
data4Scaled <- scale(data4[c(-1:-2, -6)])
pcaTestData4 <- prcomp(data4Scaled, center = TRUE, scale. = TRUE)
print(summary(pcaTestData4))

# Try with data5 after eliminating outliers
data5Scaled <- scale(data5[c(-1:-2, -6)])
pcaTestData5 <- prcomp(data5Scaled, center = TRUE, scale. = TRUE)
print(summary(pcaTestData5))

# Plotting the standard deviation of principal components
par(mfrow = c(2, 2))
plot(pcaTestData3$sdev, type = "l", ylab = "SD of PC", xlab = "PC number",
    main = "57 Variables")
plot(pcaTestData4$sdev, type = "l", ylab = "SD of PC", xlab = "PC number",
    main = "7 Variables")
```

```r
plot(pcaTestData5$sdev, type = "l", ylab = "SD of PC", xlab = "PC number",
    main = "7 Variables, no Outliers")

par(mfrow = c(1, 1))
# robData3 <- cov.rob(data3Scaled)#find the center and
# Sigma md <- mahalanobis(data3Scaled,
# center=robData3$center, cov=robData3$cov) n <-
# nrow(data3Scaled);p <- ncol(data3Scaled)
# plot(qchisq(1:n/(n+1),p), sort(md),
# xlab=expression(paste(chi^2,' quantiles')), ylab='Sorted
# Mahalanobis distances') abline(0,1) #This doesn't work as
# the x's are collinear

robData4 <- cov.rob(data4Scaled)  #find the center and Sigma
md <- mahalanobis(data4Scaled, center = robData4$center, cov = robData4$cov)
n <- nrow(data4Scaled)
p <- ncol(data4Scaled)
plot(qchisq(1:n/(n + 1), p), sort(md), xlab = expression(paste(chi^2,
    "
quantiles")), ylab = "Sorted Mahalanobis distances")
abline(0, 1)  #This works

robData5 <- cov.rob(data5Scaled)  #find the center and Sigma
md1 <- mahalanobis(data5Scaled, center = robData5$center, cov = robData5$cov)
n1 <- nrow(data5Scaled)
p <- ncol(data5Scaled)
plot(qchisq(1:n1/(n1 + 1), p), sort(md1), xlab = expression(paste(chi^2,
    "
quantiles")), ylab = "Sorted Mahalanobis distances")
abline(0, 1)

# Rotation Matrices
data3Rotation <- pcaTestData3$rotation
data4Rotation <- pcaTestData4$rotation
data5Rotation <- pcaTestData5$rotation

print(round(data3Rotation[, 5], 2))
print(round(data4Rotation[, 5], 2))
print(round(data5Rotation[, 5], 2))


# Comparing top 10 values of 5th pca for response vars in
# dataset 4 with themselves in dataset 3
# ------------------------

# Ensure row names are consistent between the two matrices
rows_to_extract <- rownames(data4Rotation)

# Extract corresponding rows from data3Rotation
filtered_data3Rotation <- data3Rotation[rows_to_extract, ]

# Combine the two for comparison
comparison <- list(data3 = filtered_data3Rotation, data4 = data4Rotation)
```

```
# Create a data frame for side-by-side comparison
comparison_df <- data.frame(Variable = rows_to_extract, data3_PC5 = round(filtered_data3Rotation[,
    "PC5"], 4), data4_PC5 = round(data4Rotation[, "PC5"], 4))

print(comparison_df)

# Lets also look at the top 10 values of the pca test of
# dataset 3
temp <- sort(abs(data3Rotation[, "PC5"]), decreasing = TRUE)[1:10]

# mod_8 <- lm(`Jump Height` ~ pcaTestData3[,'PC5'])

# Partial Least Squares Regression
# ----------------------------------------------------------------
par(mfrow = c(3, 3))

rmse <- function(x, y) sqrt(mean((x - y)^2))
data3test2 <- data3[c(7001:nrow(data3)), ]

for (i in 0:8) {
    # 9 fold cross validation
    start_idx <- i * 700 + 1
    end_idx <- start_idx + 699
    data3test <- data3[start_idx:end_idx, ]
    plsrData3 <- plsr(`Jump Height` ~ . - sex - sport, data = data3test,
        ncomp = 10, scale = FALSE, validation = "CV")  #PLSR Model for data 3
    # summary(plsrData3)
    print("Autocorrelation:")
    print(dwtest(plsrData3))  #Check for autocorrelation in ptrs model
    Sys.sleep(0.5)

    # coefplot(plsrData3, ncomp=5, xlab='Frequency')
    print("Predictors")
    print(plsrData3$loadings)
    Sys.sleep(0.5)

    plsData3 <- RMSEP(plsrData3, estimate = "CV")
    plot(plsData3, main = paste("Scree plot Cross Fold:", i +
        1))

    ypred <- predict(plsrData3, ncomp = 10)
    print(rmse(ypred, data3test$`Jump Height`))
    Sys.sleep(0.5)

    ytpred <- predict(plsrData3, data3test2, ncomp = 10)
    print(rmse(ytpred, data3test2$`Jump Height`))
    Sys.sleep(0.5)
    print("")

}

# RMSE is close on all of them so that is good Change data3
# to data5 to test with outliers
```

```r
par(mfrow = c(1, 1))

data3testFinal <- data3[1:7000, ]  #Whole training dataset

plsData3 <- RMSEP(plsrData3, estimate = "CV")
plot(plsData3, main = "Scree Plot of 10 PC's Training Data")

plsrData3 <- plsr(`Jump Height` ~ . - sex - sport, data = data3testFinal,
    ncomp = 10, scale = FALSE, validation = "CV")  #PLSR Model on whole training set
ypred <- predict(plsrData3, ncomp = 10)
print(rmse(ypred, data3testFinal$`Jump Height`))

ytpred <- predict(plsrData3, data3test2, ncomp = 10)
print(rmse(ytpred, data3test2$`Jump Height`))

# Top 10 Predictors of PLSR Model
# ----------------------------------------------------------------

rotationMatrixPLSR <- plsrData3$loading.weights
temp2 <- sort(abs(rotationMatrixPLSR[, "Comp 10"]), decreasing = TRUE)[1:10]  #Using this
temp3 <- sort(abs(rotationMatrixPLSR[, "Comp 5"]), decreasing = TRUE)[1:10]
# This hasn't produced only 1 or 2 meaningful predictors,
# so will scrap this idea

# LASSO on PCA Model
# ----------------------------------------------------------------
print(temp)
# Model with top 10 predictors from PCA test
mod_9 <- lm(`Jump Height` ~ `Relative Peak Landing Force` + `Peak Landing Force` +
    `Left Force at Peak Landing Force` + `Right Force at Peak Landing Force` +
    `Impulse Ratio` + `Peak Velocity` + `Peak Propulsive Power` +
    `Peak Relative Propulsive Power` + `Takeoff Velocity` + `Relative Propulsive Net Impulse`,
    data = data3)

design_matrix1 <- model.matrix(~`Relative Peak Landing Force` +
    `Peak Landing Force` + `Left Force at Peak Landing Force` +
    `Right Force at Peak Landing Force` + `Impulse Ratio` + `Peak Velocity` +
    `Peak Propulsive Power` + `Peak Relative Propulsive Power` +
    `Takeoff Velocity` + `Relative Propulsive Net Impulse`, data = data3)
# modeDat5_2<-mode(design_matrix1)

lmod2 <- lars(design_matrix1, data2$`Jump Height`)
lmod2
round(lmod2$Cp, 2)
par(mfrow = c(1, 1))
plot(lmod2)
# Remove `Peak Landing Force`, `Right Force at Peak Landing
# Force`,`Relative Propulsive Net Impulse`,`Relative Peak
# Landing Force`

RSS <- lmod2$RSS  #residual sum of squares for all models
df <- lmod2$df  ## number of parameters in all models
n <- dim(data3)[1]  ##number of obs.
```

```r
AIC <- n * log(RSS/n) + 2 * df  #compute AIC
BIC <- n * log(RSS/n) + log(n) * df  #compute BIC
GCV <- RSS/n/(1 - df/n)^2  #compute GCV
round(rbind(AIC, BIC, GCV), 2)

# LASSO on VIF Model
# -----------------------------------------------------------------

design_matrix2 <- model.matrix(~-`Jump Height` + ., data = data5)
lmod3 <- lars(design_matrix2, data5$`Jump Height`)
lmod3
round(lmod3$Cp, 2)
par(mfrow = c(1, 1))
plot(lmod3)

# Model from PCA selected predictors
# -----------------------------------------------------------------

mod_10 <- lm(`Jump Height` ~ `Left Force at Peak Landing Force` +
    `Impulse Ratio` + `Peak Velocity` + `Peak Propulsive Power` +
    `Peak Relative Propulsive Power` + `Takeoff Velocity`, data = data3)

summary(mod_10)
par(mfrow = c(2, 2))
plot(mod_10)  #We can conclude this model is overfitting, as the conditions are not met

termplot(mod_10, partial.resid = TRUE, pch = 16)

# Interactions
# -----------------------------------------------------------------
mod_5 <- lm(`Jump Height` ~ sex * sport * `Impulse Ratio`, data = data5)
print(summary(mod_5))

mod_6 <- lm(`Jump Height` ~ sex * sport + ., data = data5)
print(summary(mod_6))

mod_7 <- lm(`Jump Height` ~ sex * sport * ., data = data5)
print(summary(mod_7))

# Test for autocorrelation again dwtest(mod_5)
# dwtest(mod_6) dwtest(mod_7) #The tests do not work as
# they likely have too many interactions, which may affect
# the rank of the matrix These matrices are likely not full
# rank due to the amount of interactions in the model
# added.

# Hypothesis tests and ANOVA
# -----------------------------------------------------------------

summary(mod_6)  #Mod 4 is the final model, as it is the most robust and meets all criteria for hypothes
par(mfrow = c(2, 2))
plot(mod_4)
```

```r
mod_null1 <- lm(`Jump Height` ~ -sex - sport + ., data = data5)
anova(mod_null1, mod_6)  #ANOVA F-Test rejects when comparing model with no sex interaction to model wi

testPlayer <- data5[2000, ]  #Test player

predictedJumpHeight <- predict(mod_6, testPlayer)

testPlayerBBall <- testPlayer
testPlayerBBall$sport <- "basketball"
predictedJumpHeightBasketball <- predict(mod_6, testPlayerBBall)

testPlayerBBallM <- testPlayer
testPlayerBBallM$sex <- "m"
predictedJumpHeightBasketballM <- predict(mod_6, testPlayerBBallM)

comparingVals <- rbind(testPlayer$`Jump Height`, predictedJumpHeight,
    predictedJumpHeightBasketball, predictedJumpHeightBasketballM)
print(comparingVals)
```