Statistical Analysis and Visualisation of High Performance Athlete Data

David Smyth
13383861

Final Year Project
Supervisor: Dr. John Newell

February 14, 2017

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of (degree or masters) is entirely my own work and had not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ____________________ ID no: ____________________
Date: ____________________

Abstract

This project comprises of a statistical analysis and an investigation into visualisation techniques of a large data set obtained from GPS tracking units. The data originated from a competitive soccer team and was not simulated, hence challenges such as missing data, high dimensionality and possible outliers were all present when attempting to perform the analysis. The main goal of the visualistion of the data was to develop a general tool that could offer insights into player performance and training intensity based on a dataset that can be uploaded by a coach. The Shiny package in R was used to achieve this due to it's natural fit to the nature of the problem. The main goal of the statistical analysis of the data was to provide an insight to how a variable in the data, New.Bodyload, can be interpreted and reconstructed using other variables in the data that are more familiar to coaches and athletes.

TODO: PCA analysis, Tidy Regression Tree, Clustering, Sort out github & Dropbox files, Conclusions and Model Comparisons table.

Contents

Introduction

1.1 Motivation

In recent years, the vast majority of major sports teams have made the choice to set up a dedicated data analytics department in order to boost the performance of their athletes. Stastical analyses are now motivating many decisions that in the past would have been reserved solely for instinct and experience (reference?). This is due to the fact that the volume of data available to sports teams is increasing exponentially and storage capacity for this data has become a reality in recent years (provide solid state drive reference).

A new barrier has risen for sporting teams attempting to utilise this data: how can patterns in these vast amounts of data be abstracted out to give meaningful and interpretible conclusions about team performance? This is a real challenge, due to the scale of the problem. If a dataset has 70 different variables and it is desired to graphically explore the pairwise relationships of each of these variables, 70C2 = 2,415 graphs need to be examined! This is clearly unfeasible and so more sophisticated techniques need to be employed in order to systematically examine and analyse the data. This is the main motivation behind the choice of this project.

1.2 Data

I used a single dataset throughout this project that was obtained from a competitive soccer team. The data was generated from GPS devices developed by GPSports, a company involved in both the development of such devices as well as providing interpretation of the device output. GPSports report that their units typically use a 15Hz positioning sampling rate, the basic signal from which is enhanced through intelligent algorithms (rules) that use a combination of GPS signal, athlete speed, heading (direction) and activity immediately prior to the sampling point [1]. This results in a reported <1% distance error to the true distance covered [1]. This is important since many of the derived variables in the data depend on distance. It is also important to note that it is well documented that the instrumentation is liable to errors in most of the variables that it records, so all GPS data will contain some degree of inherent uncertainty due to irreducible instrumentation noise/error.

In the raw dataset, there were 8028 observations of 72 variables. The data was recorded over the period 2015/01/21 - 2016/01/16, with 129 unique training dates in that period. Each observation held the variables recorded during a distinct drill that an athelete had performed. The observations were recorded for 50 different athletes. The maximum number of unique training dates for any one athlete was 91, the minimum was 1 and the mean was 33.42. The variables recorded were diverse, including drill names, drill times, heart rate data, speeds the athlete achieved, metabolic information as well as zonal variables which provide the details of the activity within a certain threshold, for example the time spent running above 5m/s.

Statistical Analysis

2.1 Data Preprocessing

The preprocessing file for this project can be found on github at ______.The data provided for this project was in csv format, therefore a large portion of the preprocessing work was already taken care of. I chose to use R for almost all of the analysis performed in this project due to its simplicity and ease of use. Once I had read in the csv as a data frame using the read.csv function, the first step taken was to use the str() function in R to check the types of variables that R had assigned each column in the data frame. At a glance it appeared that some columns had been populated with zeros and the summary() function revealed that was the case. The columns HR.Training.Effect, Session.RPE and Sprint.Max.Acceleration were subsequently removed from the data frame. Two columns, Work.Rate.Interval.Count and Work.Rate.Interval.Duration were found to be duplicates of each other and so Work.Rate.Interval.Duration was also removed. The date column was then converted to a Date with the format d/m/y.

2.1.1 Outliers

Since New.Bodyload was the main variable of interest in this project, I began by exploring its distribution in the data set. I found one extreme outlier with a value of 14,165.74, more than 70 sample standard deviations away from the sample mean. Upon further inspection, this data point was deemed to be either a miscalibration of the gps unit or some other error since the maximum speed recorded was 101.02Km/h and most other columns contained anomylous information, and so this data point was removed. 11.1% of the data was found to lie outside ±1.5*interquartile range, which is a region associated with outliers in most analyses. These were the points that had a value of New.Bodyload>150, show in Fig 2.2. Manual inspection of these points didn't reveal any indication of miscalibration of the equipment, however they looked anomylous in the data set as seen in the density plot below and so I decided to carry out the model building process twice, with the outliers and without.

I also explored the values of New.Bodyload that were relatively small in the dataset. As noted in the data exploration section, New.Bodyload will not be zero if the unit records any acceleration with a normalised magnitude of greater than 0.25g. The GPSports site states "Accelerations above, and decelerations below the configurable threshold are reported as a events"[4]. GPSports also report that accelerations from 2g-6g are reported as low impact accelerations and that "Several thousand low intensity impacts may be seen during a football match"[4]. This gives a minimum of 11 low intensity impacts per minute for match intensity, which suggests that at least one impact will on average register for 10 every seconds of match intensity training. This indicates that there should be very few values of New.Bodyload that are 0 and so values of exactly zero may be the result of a miscalibration or other error. This is further backed up by looking at a histogram of small values of New.Bodyload in the data, where there is clearly a break in the trend for the 0-1 bin (Fig 2.1).

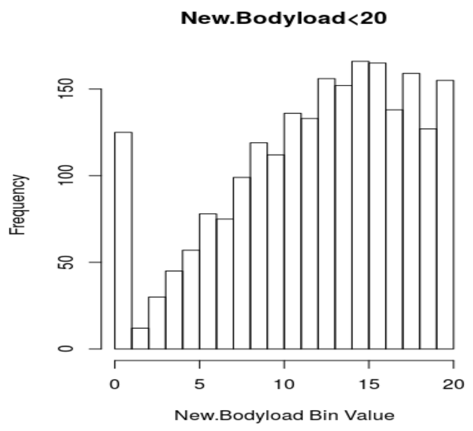


Figure 2.1: New.Bodyload small values

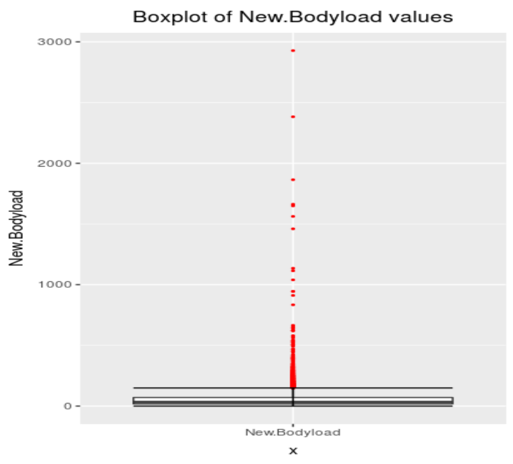


Figure 2.2: New.Bodyload large values

The other variable that contained outliers was Duration, which is fundamental to the data along with Distance. A boxplot of Duration revealed some extreme outliers with values of 641.35,609.40,603.96 and 590.90 minutes. This equates to a ten hour training session which physically seems impossib It seems much more likely that the units were accidentally left on. This is further backed up by the fact that these values were found to have occurred on the same date at the same session time. They were subsequently removed from the data set.

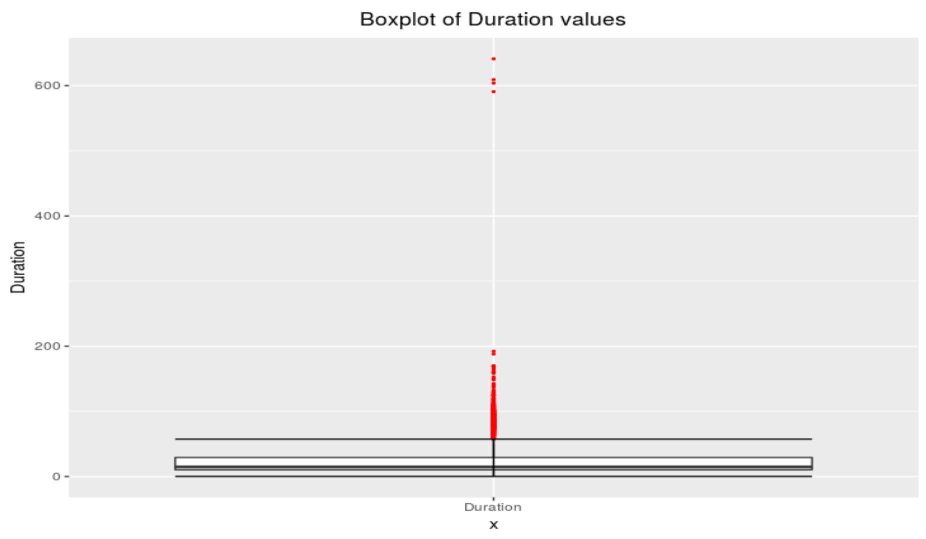


Figure 2.3: Extreme values of Duration variable

On the basis of these findings, I decided to run two analyses, one with outlier and one without because I could not fully justify removing the data containing extreme values of New.Bodyload.

2.2 Data Exploration

The data exploration began with looking up GPSports' 101 page containing a summary of how the variables in the data set are constructed [4]. An essential part of data science is understanding how the data has originated because sometimes variable names only tell half the story. The motivation of the analysis was to offer an insight into the main factors that influence the New.Bodyload variable. GPSports provide the details of how Bodyload (the old version of New.Bodyload) is derived in their FAQ [5]. They are given as follows:

1. Initialise the BODYLOAD count to 0 BL = 0.0

2. Calculate the Magnitude of the Acceleration Vector for the current acceleration sample ie. ax, ay and az.
$$V = \sqrt{ax^2 + ay^2 + az^2}$$
3. Normalise the Magnitude Vector by subtracting a notional 1G
$$NV = V – 1.0G$$
4. If the Normalised value is less than 0.25 G then go to step 2.
5. Calculate the unscaled 'BODY LOAD' (USBL) contribution for this acceleration vector as follows:
$$USBLC = NV + (NV)^3$$
6. Calculate the scaled 'BODYLOAD' taking into account the accelerometer logging rate (100Hz) and Exercise factor (EF)
$$SBLC = USBLC/100/EF$$
   Note: The Exercise Factor (EF) has been chosen so 1 hour of training activity gives a 'BODYLOAD' score of roughly. This will vary depending on the individual and sport.
7. Calculate the total 'BODYLOAD' as the accumulation of the scaled 'BODYLOAD' count
$$BL = BL + SBLC$$

The idea is that New.Bodyload is an accumulative variable that essentially aggregates the extent of the force placed on the body of a player during a training session or match. This is a useful starting point for the analysis as any models constructed should ideally explain in simpler terms the motivation of this variable as clearly the construction of the variable would not be intuitive to most people. It follows from the construction of the bodyload variable that it varies with time. This is an issue when moving to the model building phase as each measurement of New.Bodyload is taken on a different scale, namely the duration of the exercise. Treating New.Bodyload as a random variable, the consequence of this is that the observations of New.Bodyload do not come from the same distribution. DOES IT MAKE SENSE TO THINK OF THE VARIABLE IN THE DATA AS THE JOINT DISTRIBUTION OF NEW.BODYLOAD WITH TIME? THEN SHOULD 'INTEGRATING OUT' THE TIME VARIABLE GIVE A VARIABLE THAT IS IID? One way to deal with this would be to take a time series approach to the data and interpolate values in between the non-uniformly spaced data points to give uniformly spaced values for each athlete, however in practice, since the data is sparse this would not be feasible.

Treating the acceleration magnitude vector V as a random variable with some positive continuous distribution, the New.Bodyload variable can be rewritten as:

$$BL = \frac{1}{100 * EF} \sum_{i=1}^{N(t)} (V_i + V_i^3)$$

REMOVE THIS where

$$I = \begin{cases} 0 & V_i < 0.25g \\ 1 & V_i \geq 0.25g \end{cases}$$

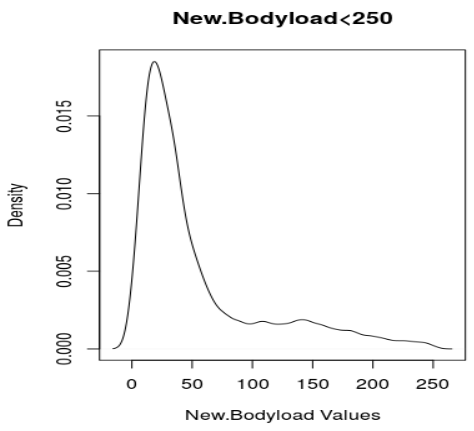


Figure 2.4: New.Bodyload Density Plot (94% of data)

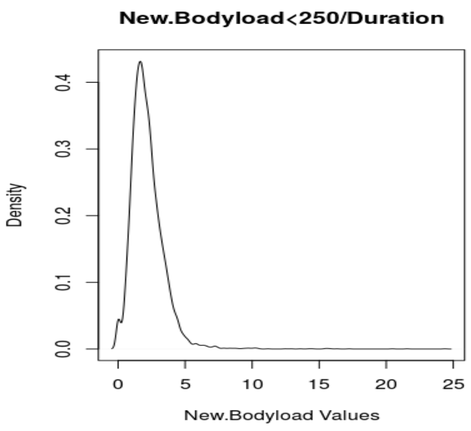


Figure 2.5: New.Bodyload/Duration Density Plot (94% of data)

Looks like this is a compound poisson process, where N(t) is a poisson random variable and $V_i+V_i^3$ is an independent (From N(t)) random variable which should be indentically distributed (for similar athletes at least). V_i should intuitively have chi distribution. Then since regression aims to minimise EPE(f)=$E[(Y-f(X))^2]$, with solution f(x)=$E[Y|X=x]$, replace Y with BL, use linearity of expectation and Wald's equation to show that what's being predicted is $\frac{1}{100*EF}E[N(t)](E[V]+E[V^3])$, where N(t) is poisson and E[N(t)] should be the inter-arrival rate. V seems like it should have chi distribution on 3 degrees of freedom, so $E[V]=$

The GPSports website mentions that it could be useful to divide each New.Bodyload observation by duration to give a new variable that might give a better comparison between athletes [4]. As seen from the summation above, this would give the sample mean of the random variable $(V_i*I-9.8)+(V_i*I-9.8)^3$, where each of the sample means should give a time-independent account of how New.Bodyload is constructed. I initially considered modelling the scaled variable, with the alternative of including the duration variable in the models as an interation term. Duration is obviously a key variable in determining the value of New.Bodyload for a player, however if it is included in a model then we can also make inference about how much it motivates the variable relative to the other variables in the model, which is useful information. This information would be lost when predicting New.Bodyload/Duration so the variable was left unscaled for the analysis. If this was a question of prediction rather than inference then both models would have been explored. Density plots of New.Bodyload and New.Bodyload/Duration are shown in figure 2.3.

From the information provided, it is clear that New.Bodyload should have some dependence on any acceleration related variables in the data. (probably get rid of this paragraph, not important) Since the New.Bodyload accumulates by $(V_i - 9.8) + (V_i - 9.8)^3$, a single 'large' value of $V_i - 9.8$ will cause the New.Bodyload variable to grow at a rate of $O((V_i)^3)$. A simulated graph of a value of BL is shown to illustrate how a single large value of V_i (possible due to a noisy measurement) can lead to an overall large value of New.Bodyload, where a high acceleration value is considered above 8g according to the GPSports website [4]. The graph was generated by sampling 100 1 minute intervals from a poisson distribution, lambda=15 (representing 15 impacts per minute, in keeping with GPSports documentation) and then scaling these observations by a corresponding sample from a N(1.6,0.2) distribution which ensures that each observation is a value of acceleration rather than an observation of an acceleration. The largest observation was then multiplied by three to show how a single erroneous record can have a large effect on an observation.

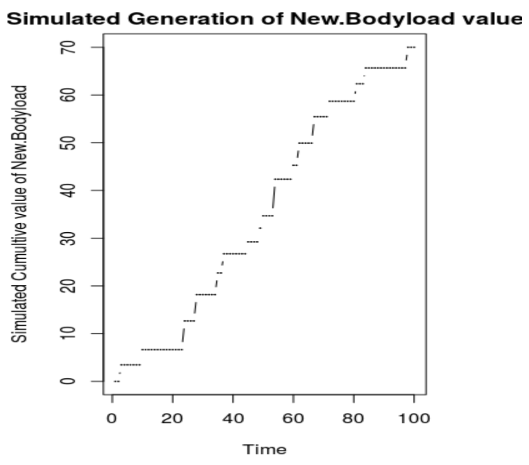


Figure 2.6: New.Bodyload Simualated Data Point

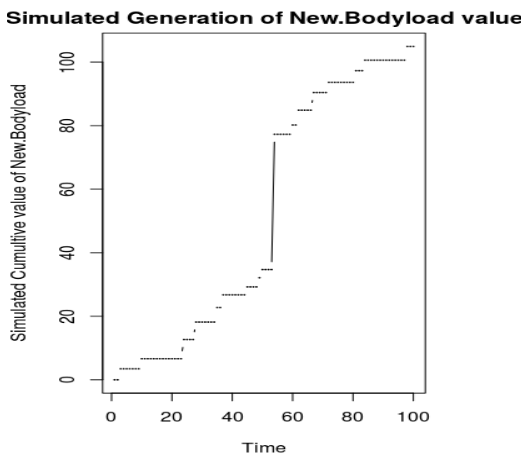


Figure 2.7: New.Bodyload Simualated Data Point with single noisy observation (3 times actual value)

Hadley Wickham's excellent book R for data science recommends the following set of steps for exploring data in a systematic method [3]:

1. Generate questions about your data.
2. Search for answers by visualising, transforming, and modelling your data.
3. Use what you learn to refine your questions and/or generate new questions.

I chose to follow these set of steps in my analysis although as the book goes on to state, exploration is as much a state of mind as a strict set of rules.

2.2.1 Duplicated/Missing Values

The first thing to check for was duplicated/missing values in the data. This was easily carried out using simple R functions.

```
TRUE%in%duplicated(anonymisedData)
[1] FALSE
naPresent=lapply(names(anonymisedData),function(x)TRUE%in%is.na(anonymisedData[,x]))
which(unlist(naPresent))
integer(0)
```

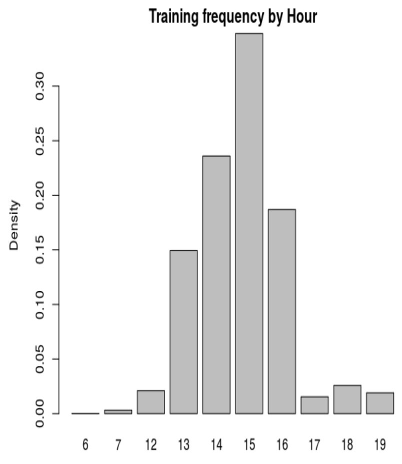
There were no duplicated values in the data and no NA values in the data. The next question to ask was if there were any missing values in the non-numeric columns.

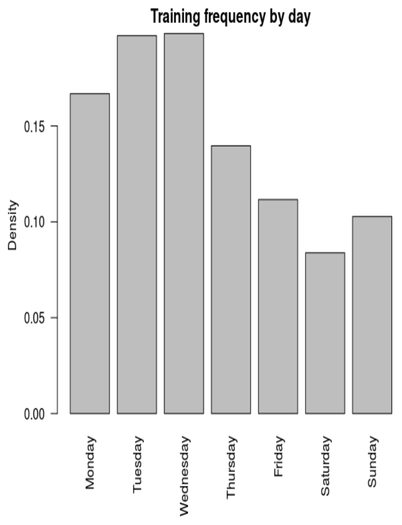
```
nonNumericCols=names(anonymisedData)[!names(anonymisedData)%in%numericCols]
missingVals=lapply(nonNumericCols,function(x)TRUE%in%(' '%in%anonymisedData[,x]))
missingVals=nonNumericCols[which(unlist(missingVals))]
missingVals
[1] "Drill"     "Position"  "Day.Code"  "Squad"
countMissingVals=lapply(missingVals,function(x) sum(' '==anonymisedData[,x]))
rbind(missingVals,round(unlist(countMissingVals)/nrow(anonymisedData)*100,3))
     [,1]       [ ,2]         [,3]       [,4]
 "Drill"    "Position"   "Day.Code"   "Squad"
"21.926"      "0.224"      "2.305"  "38.607"
```

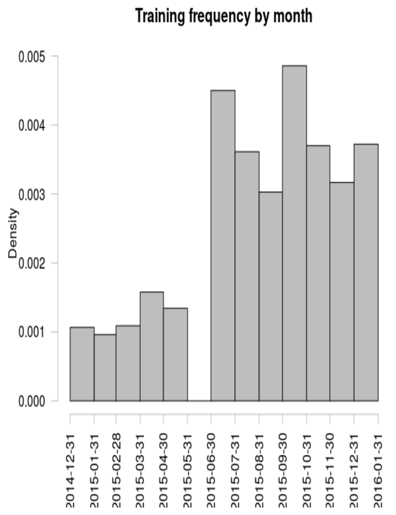
The percentages shown could be an issue if using these variables later in the analysis. The drill variable is the most notable of these as it would make sense that high intensity drills should have higher values of New.Bodyload, so the categorical drill variable could appear in models and imputing missing values may be worthwhile.

2.2.2 Training Frequency

An important part of the data exploration is the frequency at which the players train. It could also be the case that drills carried out in the morning tend to produce higher values of New.Bodyload compared to evening, for example. This can offer information about how New.Bodyload is constructed as well as visualisations of autocorrelation of variables like New.Bodyload. The density plots below shows how frequently the player train daily, weekly and monthly (non-normalised for time zone).







The plots reveal some interesting information: it seems like players are likely to train from 1pm-4pm, so it could be the case that if multiple trainings are carried out per day, New.Bodyload values will be lower in the 17:00 - 19:00 bracket. There is also a clear trend of training more often earlier in the week, which provides a sanity check if nothing else as games are usually held on Saturdays when the training frequency is lowest. The last plot arguably reveals the most information, as it appears that players train at a much higher frequency in the later months of 2015. The absence of data in June represents the players' break between the end of the 14-15 season to the beginning of pre-season training for 15-16. The last plot is worth investigating further, so I generated a plot of mean duration per month:

This shows that the average duration of training sessions was longer in the early months of 2015 but the sessions were less frequent, which makes sense. Since New.Bodyload is an accumulative variable, it would make sense that its mean value should be higher for high duration exercises over each date:

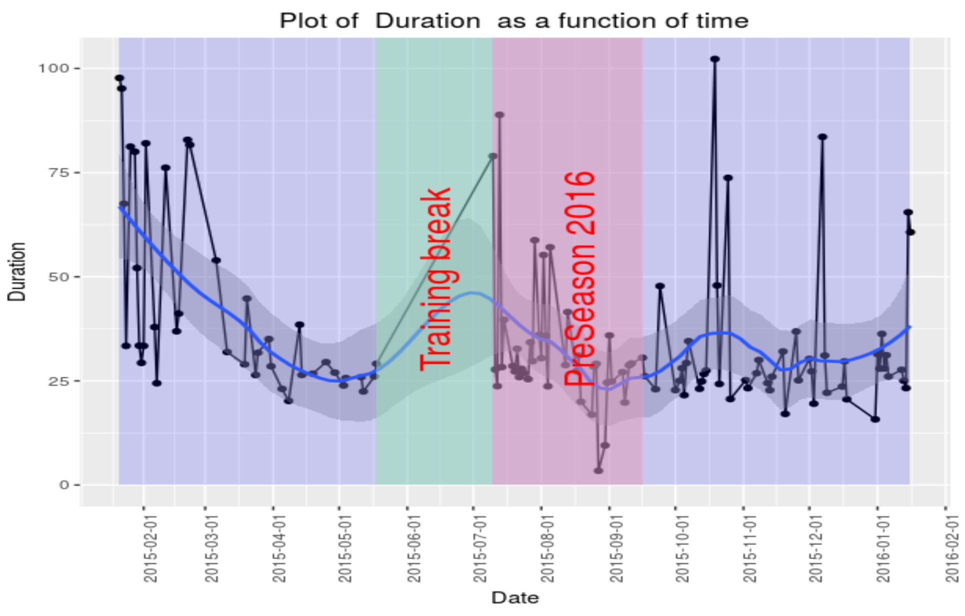


Figure 2.8: Mean Duration value per Date

This plot shows that the mean New.Bodyload on any date is higher for high duration exercises, however this information can be misleading as New.Bodyload is accumulative and so the mean value doesn't scale the variable properly (is not a sufficient statistic?) (5 observations of New.Bodyload values of 10, each over a duration of 10 minutes gives a mean value of 10, wheras 1 observation of a New.Bodyload value of 50 over a duration of 50 minutes gives a mean value of 50.). To really understand how New.Bodyload is varying over time, a plot of the accumulative value of New.Bodyload/Duration per date is required:

This plot gives a lot of information: when New.Bodyload is scaled by duration, it's mean value per date appears constant ± some noise term. This could indicate that predicting values of New.Bodyload scaled by duration could give us a date-independent variable which could simplify the analysis. It is clear from this exploration that New.Bodyload has a significant dependence on the Duration variable (high duration implies high New.Bodyload), but it may not have a dependence on the date at which it was recorded. This agrees with the GPSports documentation on the variable. (Must talk about sufficiency etc. here as not every athlete trains on each date, so if getting the sum of 20 athletes on one date and the sum of 10 athletes on another date, can't compare these values)

2.2.3 Correlations and Predictors

The next part of the data exploration focussed on exploring possible predictors of New.Bodyload. I first decided to explore non-numeric predictors. Position the first non-numeric predictor explored: it could be the case that certain positions have a tendency towards higher values of New.Bodyload. The following graph indicates that this could be the case:

Again here it made sense to first scale New.Bodyload by duration in order to be able to compare the values. (Because the scaled variables come from same distribution, unscaled don't?). The graph does not display any clear trend. Since football teams will typically train together, it makes sense that players will experience similar values of New.Bodyload and it appears this variable will not be a good predictor in any model.

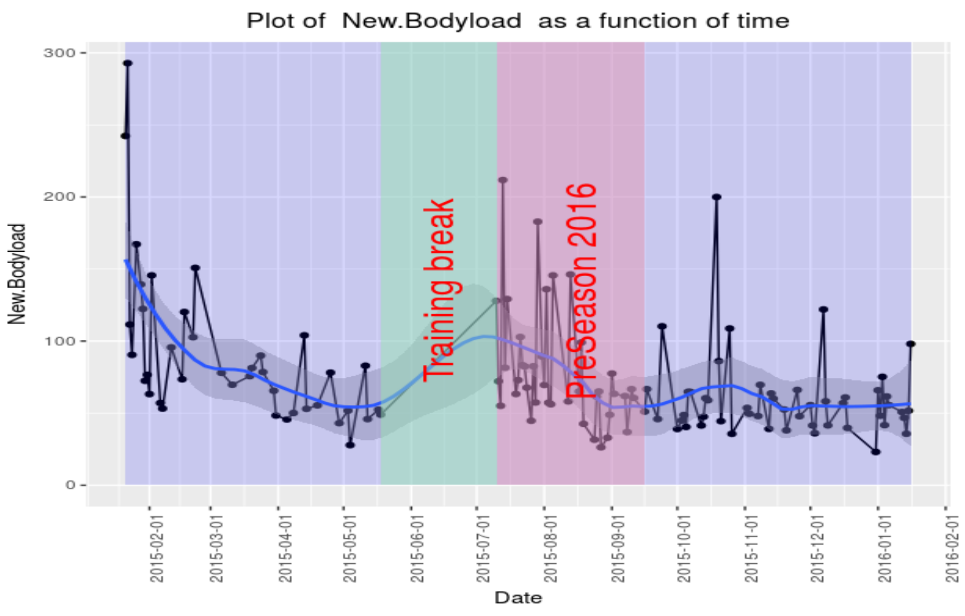


Figure 2.9: Mean New.Bodyload value per Date

Drill was the other non-numeric variable examined. As discussed in 2.2.1, 22% of values were not provided. A quick summary of the variable revealed too much ambiguity in the drill names for them to be effective predictors, for example "3 Colour Possession" and "3 Colours Possession 1/4 Pitch 7v7v7" might be distinct from each other but it is not clear if this is the case. This would lead to confusion when attempting to make inferences about the variable.

The other predictors provided in the data that were worth exploring were all numeric. I began by building a correlation plot between all of the variables. This gave a quick indication of linear associations between variables: The plot clearly shows that there a lot of strong linear associations between the variables. From this plot I found that some variables in the data could be almost fully explained by linear combinations of other variables, which lead to the removal of Equivalent.Distance, Max.HR, Sprint.Count, Work.Rate.Interval.Count and Metabolic.Load..absolute. from the data for the modelling phase, the reason being that when constructing general linear models using ordinary least squares, multicollinearity of predictors leads to the ordinary least squares estimator $\hat{\beta}_{OLS} = (X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}y$, which does not exist because $X^{\mathsf{T}}X$ either cannot be inverted or is inverted in such a way that numerically unstable estimates of the regression coefficients result (small changes in X can result in large changes to the estimated regression coefficients). From this plot, it seems like New.Bodyload has a linear relationship with a large number of variables in the model which means that a linear model should be able to explain how the variable is constructed reasonably well. It also suggests that there is strong multicolinearity between many of the possible predictors, which will need to be addressed at the modelling stage.
INCLUDE PAIRS PLOTS ONCE THE MODELS HAVE BEEN BUILT, THIS SHOWS THE LINEAR RELATIONSHIPS BETWEEN THE PREDICTORS AND THE MODEL.

Questions: Autocorrelation of variables? Effect of duration on the data? Which variables cat, which numeric? When do players train? Do players tend to train for the same amount of time. Covariations between variables? (correlation plot) Does the data have any missing values? Any duplicates? $geom_bin2d(mapping = aes(x = carat, y = price))$

Are some drills more common than others?

First, I generated a pairwise correlation

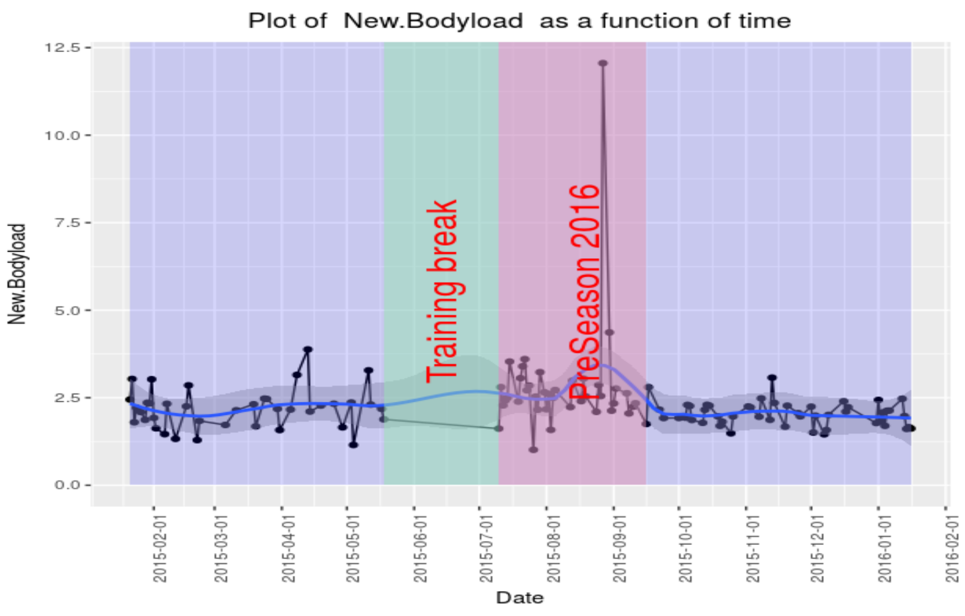


Figure 2.10: NBL scaled by Duration

2.2.4 Time Series

Many variables in the dataset depended on the time component of the data. This was mostly due to the fact that some of the variables are accumulative over time, therefore they have a strong correlation with the duration variable in the data. For the sake of simplifying the analysis, I intended to treat the observations as indpendent. Clearly this is not true, so I investigated the relationship of New.Bodyload with the time component of the data.

2.3 Data Modelling

Although the data comes from a time series, for the sake of simplicity of building statistical models I made the assumption that the observations were independent of each other (maybe this had merit from the exploration stage?). As discussed in 2.1.1, I chose to build two sets of models, one using the data with suspected outliers and the other using the data with suspected outliers removed. I also partioned my data into train and test sets of size 80% and 20% of the full data set respectively. The reason for this was to ensure that I could check that the models that I built were not overfitting the data and giving false indications of their usefulness in making inference about the data. Making the assumption that New.Bodyload (Y) is some function of the predictors (X), $Y = f(X) + \epsilon$, (where ϵ is assumed to be a random error term, which is independent of X and has mean zero), the aim of the data modelling is to find a function $\hat{f}(X)$ which approximates the true function $f(X) + \epsilon = Y \approx \hat{Y} = \hat{f}(X)$.

Ideally this model will be able to answer the following questions, (recommended in the Tibshirani & Hastie book ISLR as key questions for any inference problem)[6]:

1. Which predictors are associated with the New.Bodyload, the response variable?

2. What is the relationship between New.Bodyload and each predictor (if any)?

3. Can the relationship between New.Bodyload and each predictor be adequately summarized using a linear equation, or is a more complex relationship more informative?

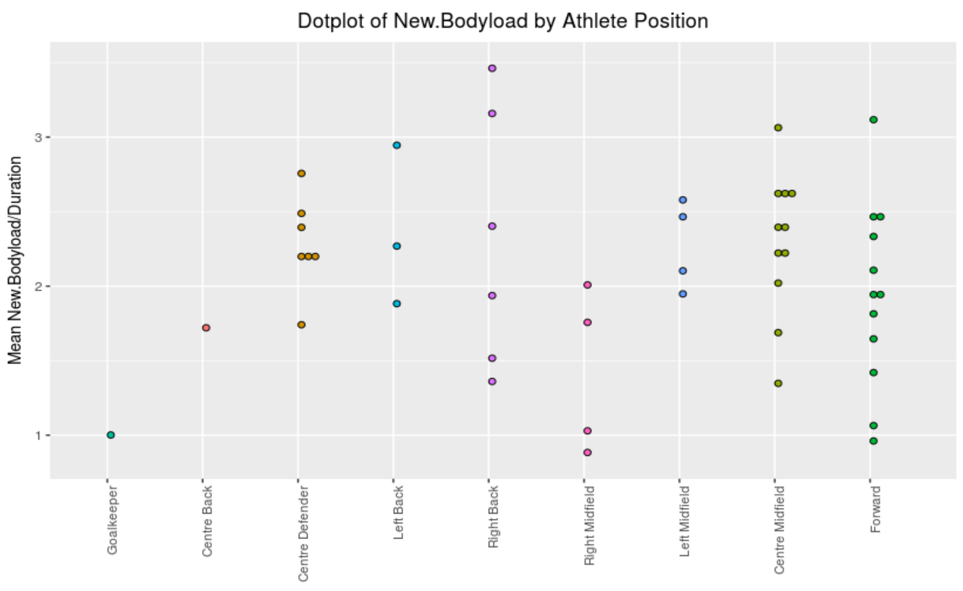


Figure 2.11: Mean NBL values of each athlete grouped by Position

2.3.1 Intended Outcomes

During the modelling stage of the data, I aimed to achieve a number of outcomes:

1. Explore Dimension Reduction Techinques. There are 60 variables in the data frame that can be used in the modelling phase, which could lead to a model that is highly predictive of New.Bodyload but difficult to interpret. Dimension Reduction of the data has the effect of limiting the number of variables used in statistical models which can help account for most of the variance of New.Bodyload while also explaining how New.Bodyload is constructed in a concise manner.

2. Based on from 1., an interpretable model is desired. This rules out highly flexible models such as random forests. The models I explored included linear models and tree based models.

3. Once the models had been constructed, I needed a method of evaluating and comparing them. R^2 [illegible]ues for linear models and Mean Squared Error usually gave a good indication but I also explored other methods of comparing models.

4. Basis functions to develop separate models for 'groups' in the data.

The data modelling process was carried out twice, once with supspected outliers and once without suspected outliers as outlined in 2.1.1. The construction and anlayis of the models discussed in the next section relate to the data with suspected outliers removed, with a section at the end discussing how the modelling process with outliers affected the analysis.

2.3.2 Linear Models

The data used for this section of my project had suspected outliers removed, totalling 93% of t[illegible] data available for this project. Linear models are useful for both prediction and inference due to their simple form, theoretical support, quick computation time, flexibility in incorporating interaction terms nad transformations of predictors, ability to include dummy variables for categorical variables and easy interpretation. Linear models are built on four main assumptions:

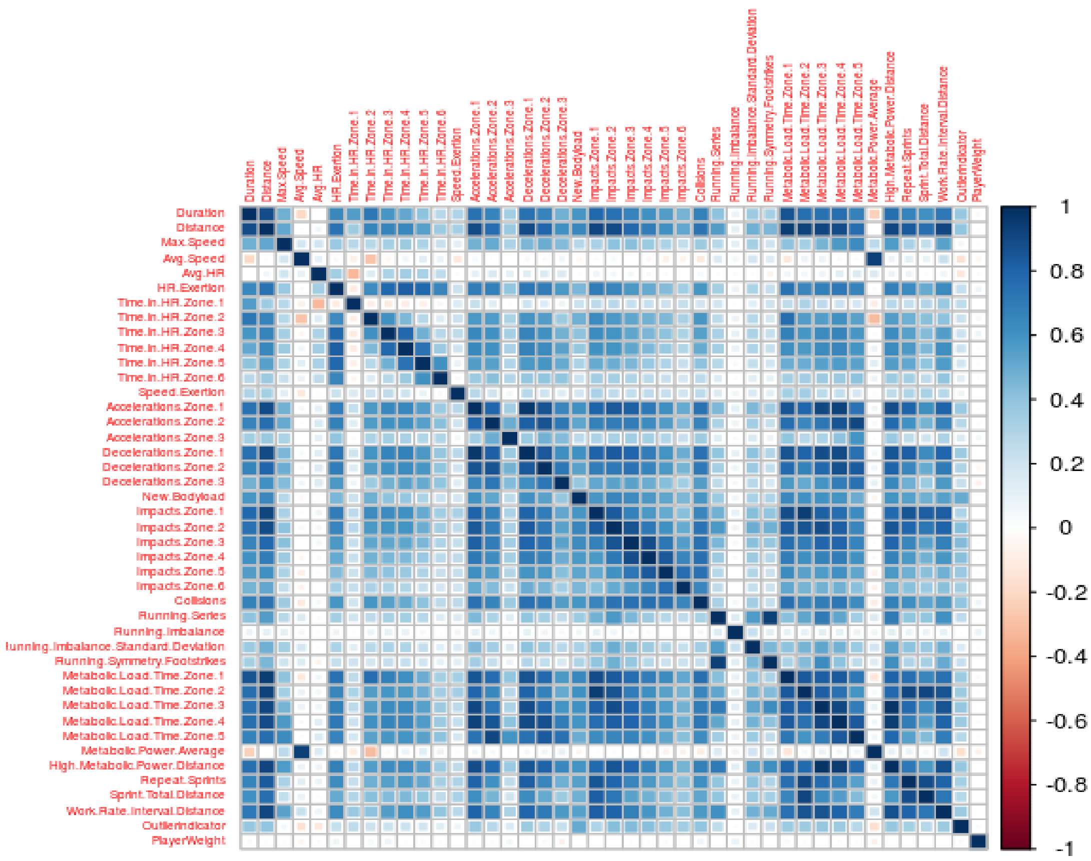


Figure 2.12: Correlation Matrix of Numeric Variables

1. Linearity of $\hat{f}(x)$. Linear models are parametric, where the functional form of f is assumed to be $f(X) = Y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n$, where each X_i is the value of the i_{th} predictor and each β_i is a scalar value which can be thought of as the weight of the corresponding predictor in estimating New.Bodyload.

2. Constant variance of residuals (homoscedasticity). Homoscedasticity implies that different response variable observations have the same variance in their residuals, regardless of the values of the predictor variables. If the residuals of the predictions did not have constant variance, then the standard error of the residuals will be biased. Since the standard error is important when conducting significance tests and calculating confidence intervals of coeffiecient values, biased standard errors result in misleading conclusions. Since linear models minimise MSE over all observations, it makes sense that the residuals produces by linear models should not be biased over some regions but not others.

3. Independence of residuals. This is especially relevant in this analysis as time series data can have seasonal effects which lead to correlated residuals. If the residuals are not independent of each other, this can mean that the linear assumption is incorrect.

4. Normality of residuals. This assumption is made so that noise in measuremen

1) Correlation matrix - when computing the matrix of Pearson's Bivariate Correlation among all independent variables the correlation coefficients need to be smaller than 1. 2) Tolerance - the tolerance measures the influence of one independent variable on all other independent variables; the tolerance is calculated with an initial linear regression analysis. Tolerance is defined as T = 1 - R^2 for these first step regression analysis. With T ¡ 0.1 there might be

multicollinearity in the data and with T ¡ 0.01 there certainly is. 3) Variance Inflation Factor (VIF) - the variance inflation factor of the linear regression is defined as VIF = 1/T. Similarly with VIF ¿ 10 there is an indication for multicollinearity to be present; with VIF ¿ 100 there is certainly multicollinearity in the sample. 4) Condition Index - the condition index is calculated using a factor analysis on the independent variables. Values of 10-30 indicate a mediocre multicollinearity in the linear regression variables, values ¿ 30 indicate strong multicollinearity.

Simple Linear Regression

The next approach I took was to fit a simple linear model. The results of fitting the linear model are shown in figure 2.12. The figure given for Multiple R-squared here represents the ratio of the explained variation of New.Bodyload to the total variation of New.Bodyload, according to the following formula:

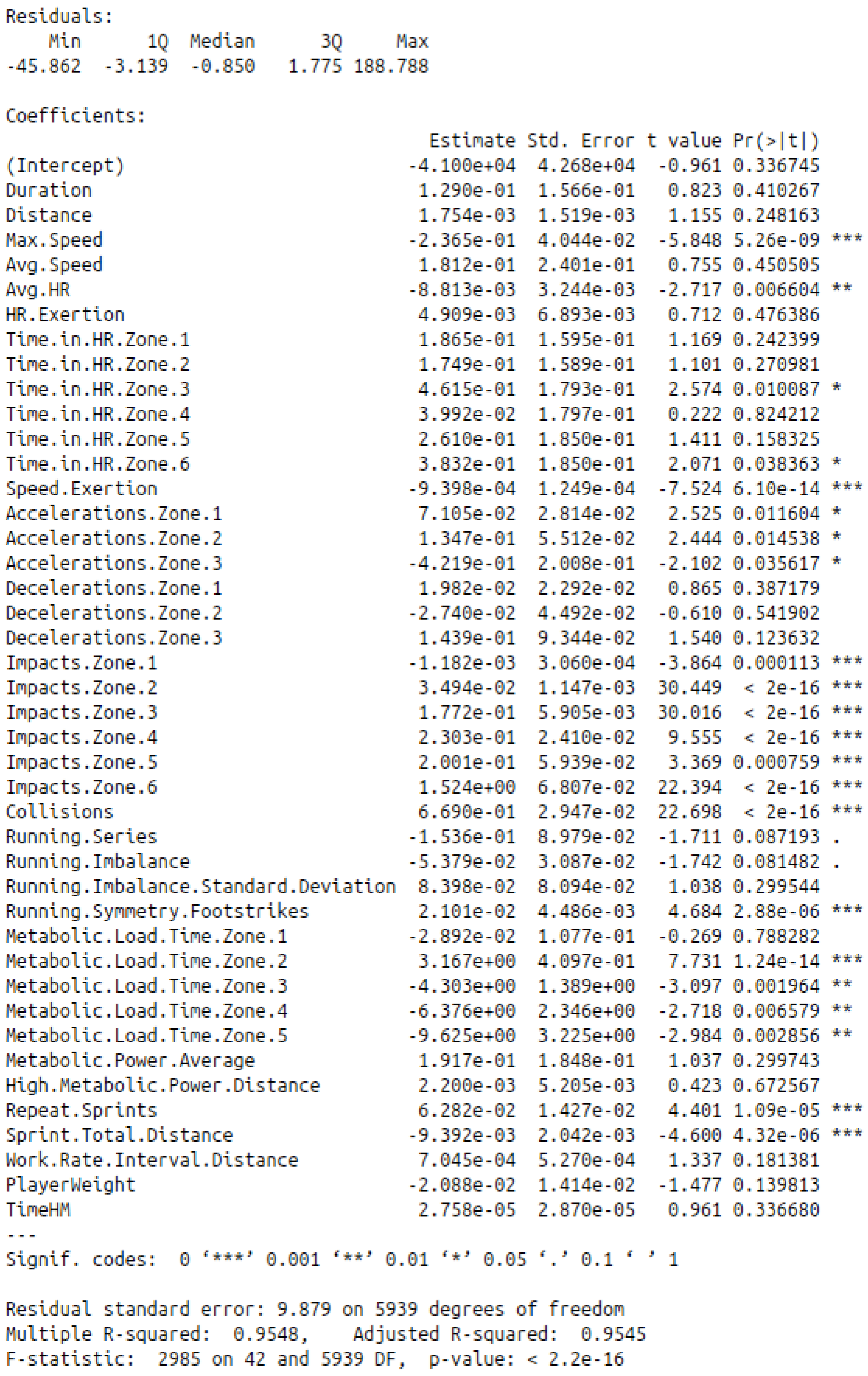
$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{1}^{n}(y_i - \bar{y})^2}$$

The high value of R^2 here indicates that the model is fitting the data well, however the residual plot associated with this model shows that the model predicts better on small values of New.Bodyload compared to large values of New.Bodyload. A normal quantile plot also indicates that the errors associated with the predictions of New.Bodyload are not normal, which means that the standard errors associated with the predictors in the model may be very inaccurate, since the calculations of standard errors and significance tests for coefficients are all based on the assumptions of homoscedasticity, independence and normality of residuals. The runDiagnostics function, found in section __________ of the appendix, was used to generate the plots shown in the figures.

Since the goal of this analysis is inference, this model is not useful because it contains too many predictors and it's coefficients cannot be accurately estimated. The model summary shows that variables like Distance and Duration are not significant in the model which strongly goes against intuition. The diagnostic plots also reveal that the model is capturing the overall trends in the data but is tending to underpredict large values. Clearly, there is a lot of room for improvement for this model, but it gives some valuable information: The impact variables are all highly significant, which follows intuition for this problem. The R^2 value also sets a benchmark for other models. Since adding predictors can only increase this value, a challenge is now presented in approaching this value using as few predictors as possible.

As discussed in section 2.2.3, the effects of multicollinearity should be explored in this problem. Each predictor weight in the model, β_i has an associated standard error, which provides a confidence interval of the coefficient value according to the equation: $Var(\hat{\beta}) = (X^TX)^{-1}\sigma^2$, given the assumption that y_i are uncorrelated and have constant variance σ^2, and that the x_i are fixed (non random)[7]. As discussed in section 2.2.3, $(X^TX)^{-1}$ will have a non-stable solution if columns of the matrix are almost linearly dependent and so the estimates of the standard errors of the predictors will be correspondingly unstable, hence it is desirable to have a quantitative sense of the amount of multicollinearity in a model. One way of measuring the effects of multicolliearity in predictive models is to calculate the variance inflation factor associated with each predictor in the model. The variance inflation factor can be thought of a measure of how much the variance of the estimated regression coefficient β_k is "inflated" by the existence of correlation among the predictor variables in the model. It can be calculated as follows:

$$\frac{Var(\beta_k)}{Var(\beta_k)_{min}} = \frac{\left(\frac{\sigma^2}{\sum_{i=1}^{n}(x_i^k - \bar{x}^k)^2}\right)\left(\frac{1}{1-(R^2)^k}\right)}{\left(\frac{\sigma^2}{\sum_{i=1}^{n}(x_i^k - \bar{x}^k)^2}\right)} = \frac{1}{1-(R^2)^k}$$



```
Residuals:
    Min      1Q  Median      3Q     Max 
-45.862  -3.139  -0.850   1.775 188.788 

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)                       -4.100e+04  4.268e+04  -0.961 0.336745    
Duration                           1.290e-01  1.566e-01   0.823 0.410267    
Distance                           1.754e-03  1.519e-03   1.155 0.248163    
Max.Speed                         -2.365e-01  4.044e-02  -5.848 5.26e-09 ***
Avg.Speed                          1.812e-01  2.401e-01   0.755 0.450505    
Avg.HR                            -8.813e-03  3.244e-03  -2.717 0.006604 ** 
HR.Exertion                        4.909e-03  6.893e-03   0.712 0.476386    
Time.in.HR.Zone.1                  1.865e-01  1.595e-01   1.169 0.242399    
Time.in.HR.Zone.2                  1.749e-01  1.589e-01   1.101 0.270981    
Time.in.HR.Zone.3                  4.615e-01  1.793e-01   2.574 0.010087 *  
Time.in.HR.Zone.4                  3.992e-02  1.797e-01   0.222 0.824212    
Time.in.HR.Zone.5                  2.610e-01  1.850e-01   1.411 0.158325    
Time.in.HR.Zone.6                  3.832e-01  1.850e-01   2.071 0.038363 *  
Speed.Exertion                    -9.398e-04  1.249e-04  -7.524 6.10e-14 ***
Accelerations.Zone.1               7.105e-02  2.814e-02   2.525 0.011604 *  
Accelerations.Zone.2               1.347e-01  5.512e-02   2.444 0.014538 *  
Accelerations.Zone.3              -4.219e-01  2.008e-01  -2.102 0.035617 *  
Decelerations.Zone.1               1.982e-02  2.292e-02   0.865 0.387179    
Decelerations.Zone.2              -2.740e-02  4.492e-02  -0.610 0.541902    
Decelerations.Zone.3               1.439e-01  9.344e-02   1.540 0.123632    
Impacts.Zone.1                    -1.182e-03  3.060e-04  -3.864 0.000113 ***
Impacts.Zone.2                     3.494e-02  1.147e-03  30.449  < 2e-16 ***
Impacts.Zone.3                     1.772e-01  5.905e-03  30.016  < 2e-16 ***
Impacts.Zone.4                     2.303e-01  2.410e-02   9.555  < 2e-16 ***
Impacts.Zone.5                     2.001e-01  5.939e-02   3.369 0.000759 ***
Impacts.Zone.6                     1.524e+00  6.807e-02  22.394  < 2e-16 ***
Collisions                         6.690e-01  2.947e-02  22.698  < 2e-16 ***
Running.Series                    -1.536e-01  8.979e-02  -1.711 0.087193 .  
Running.Imbalance                 -5.379e-02  3.087e-02  -1.742 0.081482 .  
Running.Imbalance.Standard.Deviation  8.398e-02  8.094e-02   1.038 0.299544    
Running.Symmetry.Footstrikes       2.101e-02  4.486e-03   4.684 2.88e-06 ***
Metabolic.Load.Time.Zone.1        -2.892e-02  1.077e-01  -0.269 0.788282    
Metabolic.Load.Time.Zone.2         3.167e+00  4.097e-01   7.731 1.24e-14 ***
Metabolic.Load.Time.Zone.3        -4.303e+00  1.389e+00  -3.097 0.001964 ** 
Metabolic.Load.Time.Zone.4        -6.376e+00  2.346e+00  -2.718 0.006579 ** 
Metabolic.Load.Time.Zone.5        -9.625e+00  3.225e+00  -2.984 0.002856 ** 
Metabolic.Power.Average            1.917e-01  1.848e-01   1.037 0.299743    
High.Metabolic.Power.Distance      2.200e-03  5.205e-03   0.423 0.672567    
Repeat.Sprints                     6.282e-02  1.427e-02   4.401 1.09e-05 ***
Sprint.Total.Distance             -9.392e-03  2.042e-03  -4.600 4.32e-06 ***
Work.Rate.Interval.Distance        7.045e-04  5.270e-04   1.337 0.181381    
PlayerWeight                      -2.088e-02  1.414e-02  -1.477 0.139813    
TimeHM                             2.758e-05  2.870e-05   0.961 0.336680    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.879 on 5939 degrees of freedom
Multiple R-squared:  0.9548,	Adjusted R-squared:  0.9545 
F-statistic:  2985 on 42 and 5939 DF,  p-value: < 2.2e-16
```

Figure 2.13: Linear Model Summary

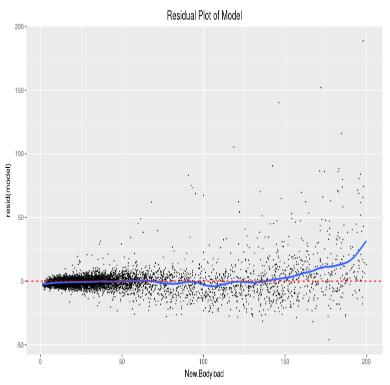


Figure 2.14: Linear Model Residuals

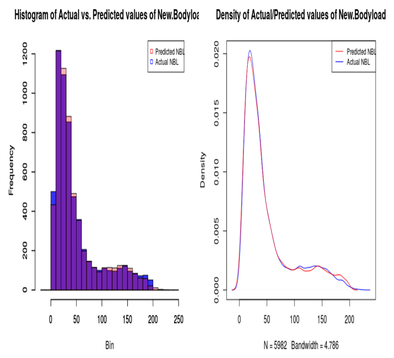


Figure 2.15: Linear Model Prediction Densities

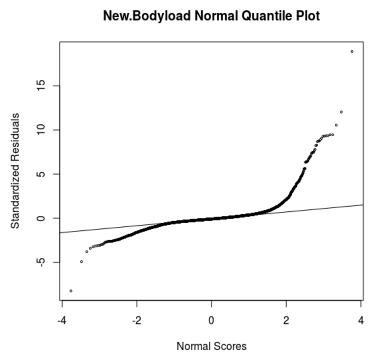


Figure 2.16: Normal Q-Q Plot

where x_i^k is the ith observation of the kth predictor and $1-(R^2)^k$ is the R^2-value obtained by regressing the kth predictor on the remaining predictors. This implies that strong linear dependence among the predictor x^k and the other predictors results in a large $(R^2)^k$ value. This further implies that large values of $(R^2)^k$ causes a large variance in β^k, which we would rather detect and avoid[8]. Note that if $(R^2)^k$=0, the variance will not be inflated at all (VIF=1), so orthogonal columns of the matrix (X^TX) are desirable for this problem.

In the simple linear regression carried out above, the vif function in the CAR package revealed that 25 out of 43 of the variables had a variance inflation factor above 10, which as a rule of thumb is considered to indicate significant multicollinearity[9].

Forward Stepwise Selection

The previous model indicated that many of the variables in the data were not predictive of New.Bodyload and their inclusion in the linear model had a detrimental effect on the confidence interval estimates of the model coefficients. Since the objective is inference, ideally the ideal model would be one that uses few variables but can explain a large amount of the variance associated with New.Bodyload. The Forward Stepwise Selection algorithm can be used to construct such a model. The idea is to limit linear regression to a maximum of p predictors, where each predictor is added to the model in a greedy manner. The algorithm runs according to the following set of steps [6]:

1. Initialise $Model_0$ as the null model, which contains no predictors.
2. For k = 0,...,p-1:
3. Fit p-k models, each of which adds one new predictor to the predictors already present in $Model_k$
4. Whichever model results in the largest reduction in RSS is set as $Model_{k+1}$
5. Select a single best model from all of the models built.

This is an heuristic algorithm, where the best solution isn't guaranteed but the runtime of the algorithm is limited. I used this technique to produce 10 models, each consecutive one having one more variable than the last. The results of the models showed some variables to be significant in the restricted model which were not significant in the Simple Linear Regression, most notably Duration. It is not clear how step 5 is achieved, since in different problems 'best' can mean different things. Here I wanted a model that was highly interpretable, so among the n models, I sought a model that could explain most of the variance associated with

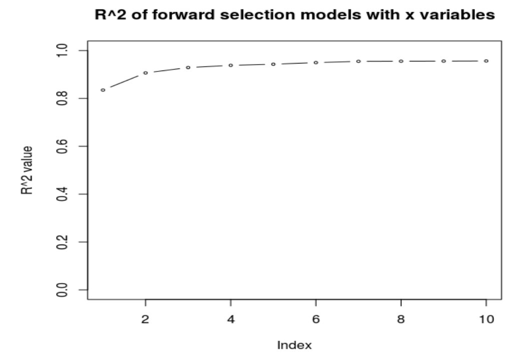


Figure 2.17: Number of predictors in model vs. R^2 value associated with model

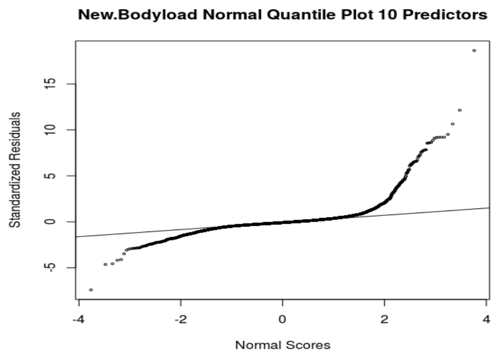


Figure 2.18: Normal Quantile Plot for model limited to 10 predictors

New.Bodyload with the fewest predictors. Figure 2.17 displays the R^2 value associated with each model containing increasing numbers of variables.

Clearly adding more than six predictors has negligible effect on increasing R^2 values. Figure 2.18 again reveals non-normal errors, which means once again that the coefficient estimates may not be accurate in the model. The top ten variables selected by the model were added in the following order: Impacts.Zone.2, Impacts.Zone.4, Duration, Impacts.Zone.6, Impacts.Zone.3, Metabolic.Load.Time.Zone.2, Collisions, Max.Speed, Speed.Exertion, Avg.Speed. Intuitively these variables all make sense in the model. The variance inflation factors for models containing 2-7 variables are shown in figure 2.19. Again residual diagnostics revealed heteroscedasticity for these models.

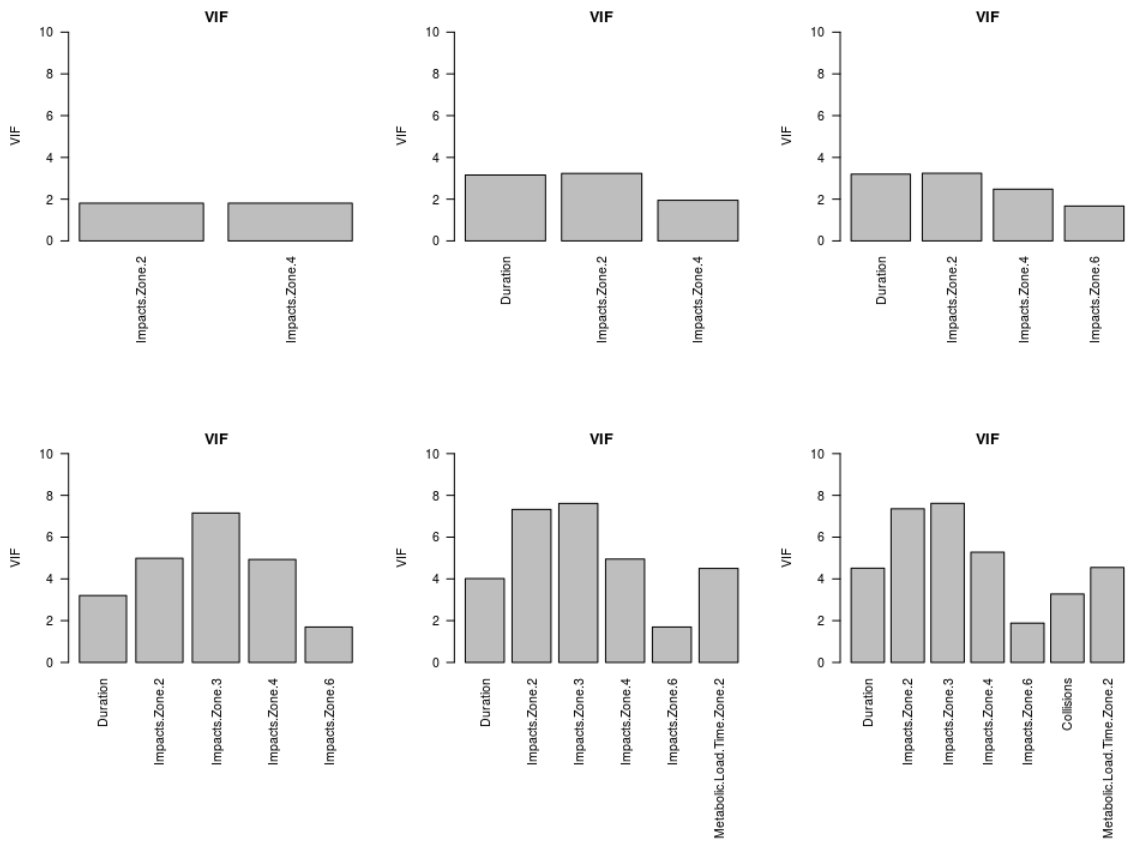
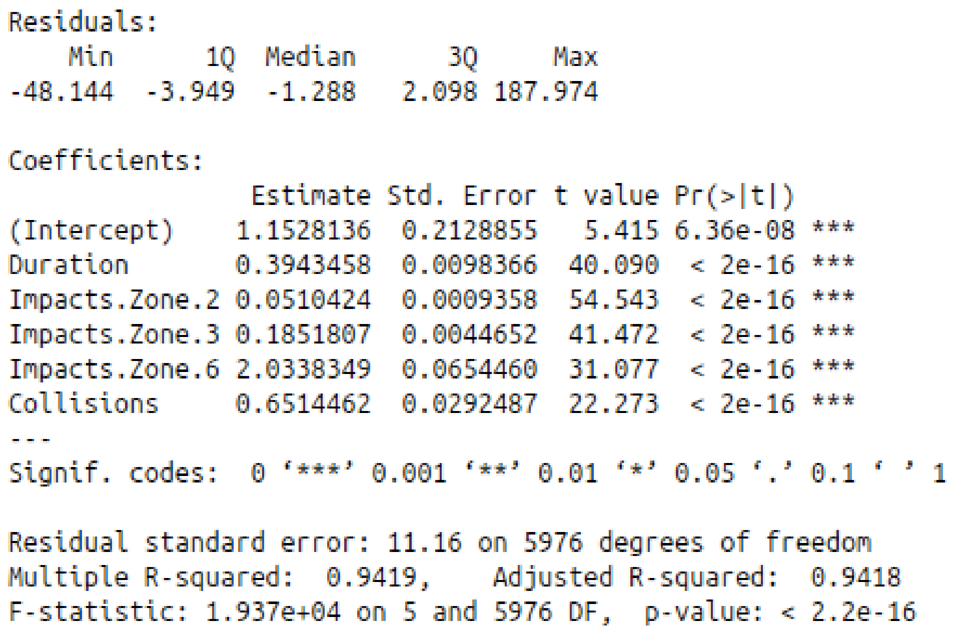


Figure 2.19: Variance Inflation factors for each Forward Selection model

Best Subset Selection

The Best Subset Selection algorithm is similar to the Forward Selection algorithm in that it chooses a subset of variables to include in a linear model. The key difference is that Best Subset Selection chooses the best linear model limited n predictors out of all possible $\binom{p}{n}$ models limited to n predictors (again the definition of 'best' is not fixed). Given that there are 43 possible predictors in the model, this task becomes unfeasibly large for values of n greater than 7 (32,224,114 models to compare). Forward Subset Selection revealed that a relatively large R^2 value can be achieved using few predictors so this technique was worth exploring. The advantage of using Forward Subset Selection with this data set was that the effects of multicollinearity were automatically mitigated by the algorithm since irrelevant variables are not included. The details of the model limited to 5 predictors using the best subset selection algorithm are shown in figure 2.19:



```
Residuals:
    Min      1Q  Median      3Q     Max 
-48.144  -3.949  -1.288   2.098 187.974 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    1.1528136  0.2128855   5.415 6.36e-08 ***
Duration       0.3943458  0.0098366  40.090  < 2e-16 ***
Impacts.Zone.2 0.0510424  0.0009358  54.543  < 2e-16 ***
Impacts.Zone.3 0.1851807  0.0044652  41.472  < 2e-16 ***
Impacts.Zone.6 2.0338349  0.0654460  31.077  < 2e-16 ***
Collisions     0.6514462  0.0292487  22.273  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.16 on 5976 degrees of freedom
Multiple R-squared:  0.9419,	Adjusted R-squared:  0.9418 
F-statistic: 1.937e+04 on 5 and 5976 DF,  p-value: < 2.2e-16
```

Figure 2.20: Best Subset Selection 5 variables

Diagnostic plots once again revealed that while the model captured the overall trend, there was heteroscedasticity present.

Lasso

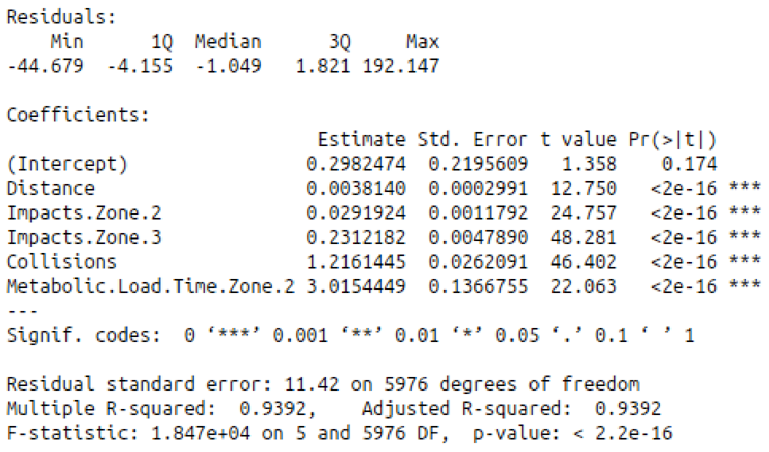
Another linear model explored was the Lasso. This model is an extension of Simple Linear Regression, where the coefficients are fit by minimising the quantity

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

where high values of λ tend to penalise coefficients with relatively high values. The idea is that a well chosen lambda value shrinks some of the coefficient values to zero while leaving the coefficients that are more efficient at reducing the RSS as non-zero valued. Further details of the lasso can be found in Tibshirani and Hastie's book ISLR [6]. As the value of lambda was varied, variables were introduced by the lasso in the following order: Distance & Impacts.Zone.2 (Simultaneously), Impacts.Zone.3, Collisions, Duration, Impacts.Zone.4, Metabolic.Load.Time.Zone.2, Impacts.Zone.6, Impacts.Zone.5, Metabolic.Load.Time.Zone.1.

Polynomial Regression

Polynomial regression was implemented to explore the possibility that there were strong non-linear relationships between some of the predictors and New.Bodyload. Diagnostic plots from previous techniques indicated that this likely wasn't the case but for the sake of thoroughness I decided to run a model to investigate. This involved creating a new data set containing non-linear transformations of each of the variables. R is suited to running transformation like this and the functions found here ________ were used to create an additional 336 variables in the model. For each predictor column vector X_j (apart from New.Bodyload), 7 new columns were created corresponding to $X_j^i, i \in \{-4, -3, -2, -1, 2, 3, 4\}$. First I carried out the forward selection algorithm to limit the runtime because the search space now contained 338 predictors rather than 43. Variables were added in the following order: Impacts.Zone.2, Impacts.Zone.4, Duration, Impacts.Zone.6, Impacts.Zone.3, Metabolic.Load.Time.Zone.2, Collisions, Duration^4, Speed.Exertion, Max.Speed^{-1}. A best subset selection limited to a maximum of 5 variables yielded the model in figure 2.20.



```
Residuals:
    Min      1Q  Median      3Q     Max
-44.679  -4.155  -1.049   1.821 192.147

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               0.2982474  0.2195609   1.358    0.174
Distance                  0.0038140  0.0002991  12.750   <2e-16 ***
Impacts.Zone.2            0.0291924  0.0011792  24.757   <2e-16 ***
Impacts.Zone.3            0.2312182  0.0047890  48.281   <2e-16 ***
Collisions                1.2161445  0.0262091  46.402   <2e-16 ***
Metabolic.Load.Time.Zone.2 3.0154449  0.1366755  22.063   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.42 on 5976 degrees of freedom
Multiple R-squared:  0.9392,    Adjusted R-squared:  0.9392
F-statistic: 1.847e+04 on 5 and 5976 DF,  p-value: < 2.2e-16
```

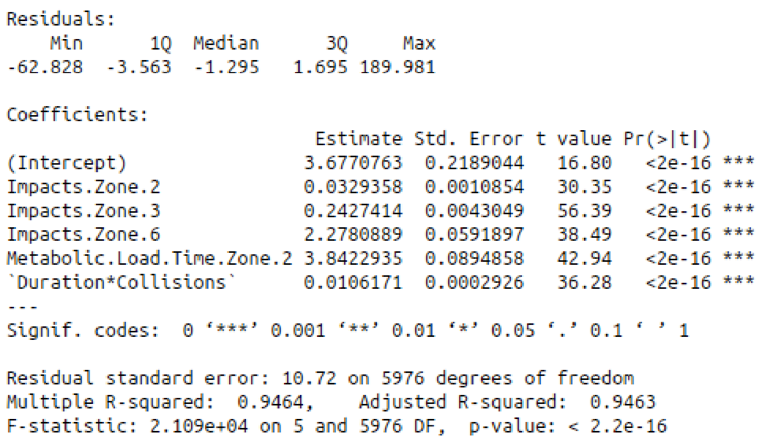
Figure 2.21: Best Subset Selection With Polynomial Terms

This was not surprising as scatter plots of New.Bodyload did not indicate non-linear relationships with any of the variables flagged as strong predictors by previous models. The anova function in R, with null hypothesis stating that the two models fit the data equally well, and the alternative hypothesis stating that the model containing polynomial terms is superior showed that each forward selection model using x variables with polynomial predictors is not superior to the forward selection model using the linear predictors.

It could have been the case that a different type of non-linear transformation could have yielded better results, however as previously mentioned, when constructing a model with the purpose of inference a linear model has the advantage of being highly interpretable. The project could have been extended by further exploring non-linear transformations.

Linear Regression With Interaction Terms

Since some of the predictors in the data set were accumulative with time, their inherent reliance on Duration implied that there could be some interaction effects in the data that could be taken advantage of. Linear models discussed in previous sections all had a coefficient β_j associated with the j_{th} predictor, which represented the change in New.Bodyload corresponding to a unit increase in the predictor value, keeping all other predictors fixed. In some regression problems, it has been observed that there are synergetic effects between predictors in the model. It could be the case, for example, that increasing values of Duration could in turn increase the effect of Impacts on an athlete, so that values of New.Bodyload increase at non-constanct rate per unit change in Collisions. In order to explore this, the variables that appeared in the best subset selection limited to ten variables were multiplied together to generate 66 new columns. Then best subset selection was ran with a maximum of 5 variables which yielded the model in figure 2.22.



```
Residuals:
    Min      1Q  Median      3Q     Max 
-62.828  -3.563  -1.295   1.695 189.981 

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 3.6770763  0.2189044   16.80   <2e-16 ***
Impacts.Zone.2              0.0329358  0.0010854   30.35   <2e-16 ***
Impacts.Zone.3              0.2427414  0.0043049   56.39   <2e-16 ***
Impacts.Zone.6              2.2780889  0.0591897   38.49   <2e-16 ***
Metabolic.Load.Time.Zone.2 3.8422935  0.0894858   42.94   <2e-16 ***
`Duration*Collisions`       0.0106171  0.0002926   36.28   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.72 on 5976 degrees of freedom
Multiple R-squared:  0.9464,	Adjusted R-squared:  0.9463 
F-statistic: 2.109e+04 on 5 and 5976 DF,  p-value: < 2.2e-16
```

Figure 2.22: Best Subset Selection With Interaction Terms

The interaction term Duration*Collisions can be interpreted as follows: for every minute that the drill was carried out for, each collision experienced in the drill over the duration of that minute increased the value of New.Bodyload by 0.0106171 ($\pm$0.0002926).

Principal Components Analysis

Principal components analysis is a technique which is used to reduce the dimensionality of the data.

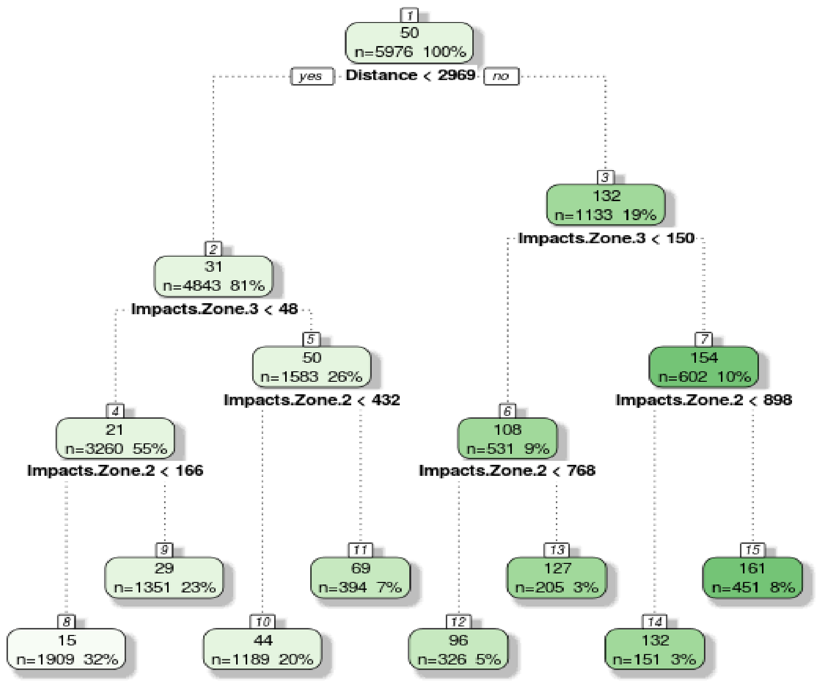
2.3.3 Non-Linear Models

Regression Tree

I began by building a simple regression tree, as tree-based methods tend to be simple and useful for interpretation. The rattle package contains the FancyRPartPlot function which helps visualise the decision tree. Decision trees are constructed by stratifying the feature space into distinct regions which have piecewise constant predicted values of New.Bodyload. The tree algorithm performs recursive binary splitting by first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ maximises the resulting reduction in RSS due to the split. This algorithm is greedy and is not guaranteed to find the best solution but it limits the search space in such a way that a solution can easily be found. The figure shows the structure of the decision tree where logical no evaluations lead to the right and yes evaluations lead to the left.

The numeric value at the top of each node displays the predicted New.Bodyload value for the partition created by the node and the bottom values represent the number of observations in that partition and the proportion of observations in that partition. The condition surrounded by *yes* and *no* is the predictor and cutpoint that further partitions the predictor space into regions that maximises the reduction in RSS across all predictors. Here clearly zone variables dominate the model once the distance variable has partitioned the data. This makes sense as impacts are a source of acceleration and the tree suggests that high distances and high numbers of impacts lead to large values of New.Bodyload.

Figure 2.23: Regression Tree generated using the rpart package



Clustering

Plots of the variables that linear models found to be predictive of New.Bodyload seemed to all have a trend of being linear up to a certain point and then have a non-linear relationship beyond that. Figures x-y show this trend. This was motivation to implement a clustering algorithm on the data to explore if there was a natural split in the data. I decided to run a K-means clustering algorithm to explore if mathematically the data would be most efficiently partitioned in correspondance with the linear/non-linear trends in figures x-y. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The K-means algorithm attempts to partition the data into K groups, C_1,..., C_k according to the criteria:

$$minimize \sum_{i=1}^{K} W(C_i)$$

with the weight function W defined using the sum of squared Euclidean distances between points by

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,j \in C_k} \sum_{l=1}^{p} (x_i^l - x_j^l)^2$$

where C_i is the i_{th} set in the partition and x_i^l represents the i_{th} observation of the l_{th} variable. The intention was for the model to split the data into two groups, where a linear model could be fit separately for each group. The results of the partition are shown for variables that are key in linear models.

2.3.4 Modelling data with suspected outliers

2.4 Results

Data Visualisation

Having completed the statistical analysis of the data, a natural question arises: how is it possible to convey the results observed in the analysis to coaches and athletes so that they can obatain an understanding of the data that they have collected. Visualisation is an important part of any data analysis as it can offer the most insight into the data due to the inherent human dependency on vision for intepretation. Typically datasets will have multiple dimensions and facets, so in order to gain a broad insight into the data, multiple graphs and images are necessary. Any data visulisations will ideally be portable, system independent and reusable. Options: Building a full website to display images(RoR, Django), Spyre (python), Shiny. (Give benefits & drawbacks of each). The obvious choice here was Shiny.

3.1 Shiny

Shiny is an open source R package that provides a high level web development framework. Shiny was developed with the aim of converting analyses into interactive web applications with minimal knowledge of HTML, CSS, Javascript and other common web-development languages. The focus is strongly on presenting graphics to the end user which can enlighten them about their data. The advantages of using Shiny are:

- It's interactive which means it can offer a dynamic user experience.
- Shiny applications are written in R and can utilise all of the libraries that are already available in the R language.
- The development of a shiny app is centred around displaying graphics generated with R to the user. If you can program with R, you can program with Shiny.

Since R was used to perform the statistical analysis, it was a natural extension to develop the visulistions using the Shiny package. Every Shiny application consists of a minimum of a server function and a user interface function which follows the "separation of concerns" design principle. The server is concerned with data manipulation and mapping user inputs to various types of output, wheras the user interface is concerned with data presentation. I followed the convention in Shiny of creating a project folder and then writing R files named server.r, ui.r and a www folder which contains css and image files used by the browser in the app. (show folder structure)

3.2 Functionality

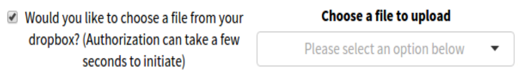
In the application that I developed, I set out a certain amount of functionality that would be necessary to make the app straightforward for the user to operate. The following subsections offer an exposition of how this was achieved.

3.2.1 Dropbox File Upload

It made sense for the app to allow the user to upload their own csv file in order to display their data. This was easily be achieved through using a local file uploader, which Shiny provides using the fileInput widget. Often sports teams will avoid storing data on local machines as it is a source of data leakage as that means the sports team's data is located on a number of different machines rather than on a central repository. It also means that if one of the machines is damaged, that data could be lost permanently. For these reasons, tools like dropbox are used to store and access athlete data. I decided to add the option to use data from the users' dropbox account to upload to the app. This required using the package rdrop2 to facilitate the user logging into their dropbox and permitting the application to upload data from their dropbox to the app. This was achieved using the following code segment:

```
dropboxDown=reactive({
  if(input$dropboxDownload){
  #only run this authorisation once (not sure how to do this without the use of global variable)
    if(!alreadyVisited){
      #consider adding new_user=TRUE
      dropbox_credentials=drop_auth(cache=FALSE)
      updateCheckboxInput(session, "dropboxDownload",label='Choose a file:')
      #update file choices
          updateSelectizeInput(session,'dropboxFileSelection',choices=basename(
          drop_search('.csv')[order(drop_search('.csv')$modified),]$path))
          #set global alreadyVisited variable to TRUE as this behaviour should only occur once
          alreadyVisited<<-TRUE
        }
        #update the input file to the file selected by the user
        inFile1=input$dropboxFileSelection
        #check file exists
        if (is.null(inFile1)){
          print('No file provided')
          return(NULL)
        }
        else{
          #can deal with RData and csv formatted files
          if(length(grep('.csv',inFile1))!=0){
            #read csv by finding the path in dropbox, ensure ProcessedData is global
            ProcessedData<<-drop_read_csv(drop_search('.csv')[match(inFile1,basename(
            drop_search('.csv')$path)),]$path,stringsAsFactors=FALSE)
            #be careful as write.csv adds an x column if column.names=FALSE
            ProcessedData$Date=as.Date(ProcessedData$Date)
          }
          else if(length(grep('.RData',inFile1))!=0){
            load(inFile1$datapath)
            ProcessedData<<-ProcessedData
          }
    }
    if(exists('ProcessedData')){
          #If the data exists, preprocess it
      Preprocess()
      #now allow the user to download a report
      shinyjs::enable("report")
      shinyjs::enable("refreshReport")
      #call data preprocessing function, look into passing Preprocess to this function
    }
  }
})
```

This led the user through the sequence of events displayed in figures 3.1-3.4:

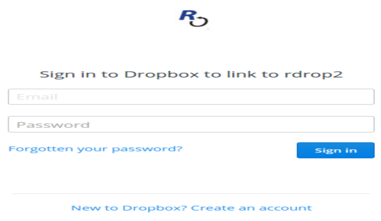


☑ Would you like to choose a file from your dropbox? (Authorization can take a few seconds to initiate)

Choose a file to upload

Please select an option below ▾

Figure 3.1: 1). Click checkbox



Sign in to Dropbox to link to rdrop2

Email

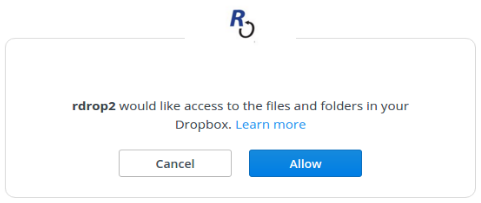
Password

Forgotten your password?

Sign in

New to Dropbox? Create an account

Figure 3.2: 2). Sign in to Dropbox

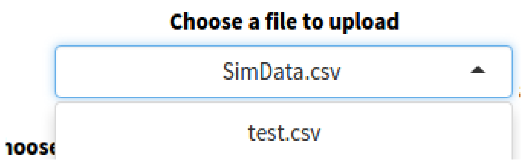


rdrop2 would like access to the files and folders in your Dropbox. Learn more

Cancel

Allow

Figure 3.3: 3). Authorise file access



Choose a file to upload

SimData.csv ▴

test.csv

Figure 3.4: 4). Upload Dropbox file

The rDrop2 package stores tokens containing the users password and username so that the user does not have to log in every time the app is used, however this caused issues. There are options in the function drop_auth which prevented the caching of login details by the application and also the option of specifying whether a new user is logging in (and whether previously cached details should be overwritten). In practice however, it seemed that the application created a file named .httr-oauth containing the cached user login information regardless of these options. This would be a major issue if the code was used in production but since the application was proof of concept, the issue was not resolved.

3.2.2 CSV Upload Format

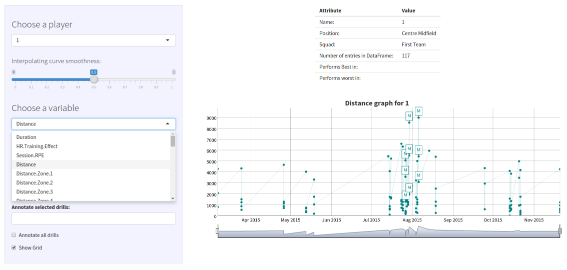
The user must upload the data in a certain file format for this application to be able to process the file. From the outset I intended to write an application that would impose a minimum amount of restrictions on the user with the intent of having a scaleable and extendable program. The modularity of the code has the consequence of decoupling the displays from the variables that are being displayed, where in the application functions took as input the name of any numeric column in the data and outputted graphs. This meant that the only variables that must necessarily be present in the data are Date and Name (of athlete). The consequence of this is that the application can be modified in a matter of minutes to display the same graphs and summaries for a csv file containing any type of numeric data.

3.2.3 Application Tabs

The rest of the functionality of the application was modularised into tabs which could be navigated in order to display a specific plot or summary.

Longitudinal Plot

The tab that the application opens on is the longitudinal plot. This plot gives the user graphical information about how some aspect of the athletes' training has changed over time. The user can subset the graph to a certain date range and has the option to compare another athlete on the same graph. The option is also given to annotate drills so that if there is an unusual looking data point, the application user can immediately figure out which training session the ususal data originated from. The R dygraph package was the main tool used to achieve this.

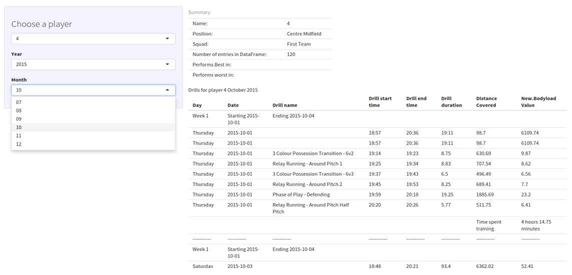


Attribute	Value
Name:	1
Position:	Centre Midfield
Squad:	First Team
Number of entries in DataFrame:	117
Performs Best in:	
Performs worst in:	

Figure 3.5: Longitudinal Plot Tab

Training Calendar

The training calendar tab provides the user with a readable account of an individual athlete's training schedule over a month in the data provided. The calendar displays which drills the athelete performed, the drill start and end times, the total duration of the drill, the distance that the player covered in the drill as well as the New.Bodyload value generated for that drill. The calendar is broken into weeks of the month



Summary

Name:	4
Position:	Centre Midfield
Squad:	First Team
Number of entries in DataFrame:	120
Performs Best in:	
Performs worst in:	

Drills for player 4 October 2015

Day	Date	Drill name	Drill start time	Drill end time	Drill duration	Distance Covered	New.Bodyload Value
Week 1	Starting 2015-10-01	Ending 2015-10-04					
Thursday	2015-10-01		18:57	20:36	19:11	98.7	6109.74
Thursday	2015-10-01		18:57	20:36	19:11	98.7	6109.74
Thursday	2015-10-01	3 Colour Possession Transition - 6v2	19:14	19:23	8.75	630.69	9.87
Thursday	2015-10-01	Relay Running - Around Pitch 1	19:25	19:34	8.83	707.54	8.62
Thursday	2015-10-01	3 Colour Possession Transition - 6v3	19:37	19:43	6.5	496.49	6.56
Thursday	2015-10-01	Relay Running - Around Pitch 2	19:45	19:53	8.25	689.41	7.7
Thursday	2015-10-01	Phase of Play - Defending	19:59	20:18	19.25	1885.69	23.2
Thursday	2015-10-01	Relay Running - Around Pitch Half Pitch	20:20	20:26	5.77	511.75	6.41
						Time spent training	4 hours 14.75 minutes
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------
Week 1	Starting 2015-10-01	Ending 2015-10-04					
Saturday	2015-10-03		18:48	20:21	93.4	6362.02	52.41

Figure 3.6: Training Calendar tab

so that athletes and coaches have a week-by-week account of the drills that were performed. The code was written so more details about the training session can be conveniently added.

Barplot

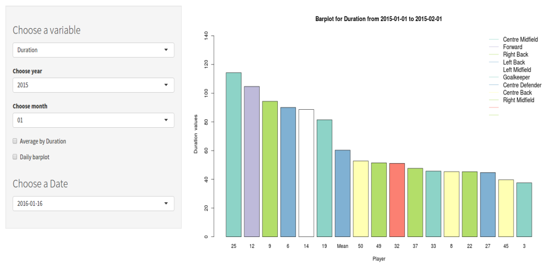
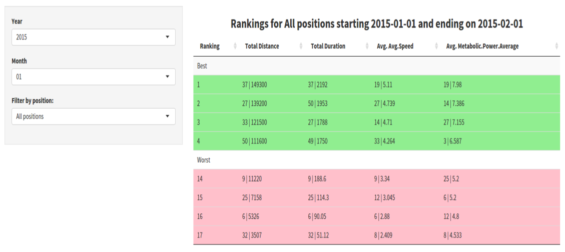


Figure 3.7: Barplot tab

The barplot tab was designed to give a graphical representation of how players in the team performed relative to each other over a date range for a certain variable. This is useful as sometimes absolute figures can lose meaning when comparing each player in the team to other players in the team quickly tells the athlete and coaches who is underperforming in certain aspects. The barplot is coloured by position so that it is easy to compare players competing for the same position. There is an option to average the plot by duration because, as discussed in 2.2.2, accumulative variables are not measured according to the same scale when mean values are taken. Therefore this plot can be somewhat misleading and should be used with caution. There is also an option to view the data with greater granularity by selecting to display the barplot over an individual date.

Rankings



Year: 2015
Month: 01
Filter by position: All positions

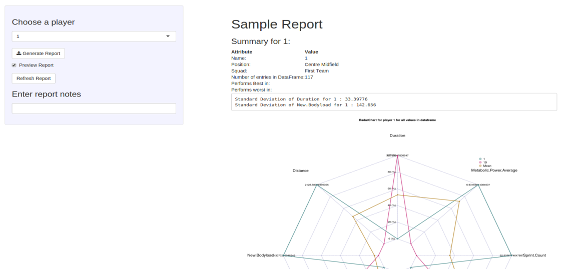
Rankings for All positions starting 2015-01-01 and ending on 2015-02-01

Ranking	Total Distance	Total Duration	Avg. Avg.Speed	Avg. Metabolic.Power.Average
Best				
1	37 | 149300	37 | 2192	19 | 5.11	19 | 7.98
2	27 | 139200	50 | 1953	27 | 4.739	14 | 7.386
3	33 | 121500	27 | 1788	14 | 4.71	27 | 7.155
4	50 | 111600	49 | 1750	33 | 4.264	3 | 6.587
Worst				
14	9 | 11220	9 | 188.6	9 | 3.34	25 | 5.2
15	25 | 7158	25 | 114.3	12 | 3.045	6 | 5.2
16	6 | 5326	6 | 90.05	6 | 2.88	12 | 4.8
17	32 | 3507	32 | 51.12	8 | 2.409	8 | 4.533

Figure 3.8: Example reactive expression [2]

The rankings tab was designed to display information about some key variables in the data in a compact and succinct manner. The figure was generated using the DT package in R. The application user is given the choice to subset the time period to a month and the players who achieved the best and worst values in each key variable are highlighted. The name of the player is displayed beside the value that the player achieved for the variable over the time period. Again, care must be taken when using accumulative variables as the means of theses variables cannot accurately be compared.

Report Downloader



Sample Report

Summary for 1:

Attribute	Value
Name:	1
Position:	Centre Midfield
Squad:	First Team
Number of entries in DataFrame:	117
Performs Best in:	
Performs worst in:	

```
Standard Deviation of Duration for 1 : 33.39776
Standard Deviation of New.Bodyload for 1 : 142.656
```

Figure 3.9: Report Download tab

The report Downloader tab facilitates the user entering some notes about a player and then download a custom report for that player. There is an option to view a rough preview of the report before downloading it, as is shown in the image. The report itself was generated using RMarkdown and introduces significant overhead to the app. A caching system for the reports or a separate report generating mechanism that runs in parallel to the application would be desirable for this tab, however to achieve this would take a considerable amount of effort. There are also security issues over downloading the report and the mechanism for storing the reports is currenlty flawed, meaning this is the only part of the application that would have issued if deployed to a server.

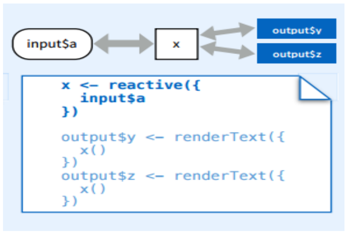
3.3 Dependencies

Shiny's key strength is it's ability to display reactive output. This is facilitated using a series of outputs generated by the server function, which will automatically update when any of the inputs that the output relies on changes. This is referred to as dependency in a Shiny application. Dependencies are automatically created by the framework in order to track when an input variable changes so that the corresponding output remains up to date. The only requirement is that each input and output has a unique identifier in the Application namespace. The dependencies reflect the fundamental relationship between outputs and inputs. For example, in my application the player summary tables changed whenever the selected player changed, hence a dependency was set up on selected player. However, the programmer must be careful to only re-run code when necessary, otherwise a large overhead would be introduced making the program sluggish and frustrating to use. Shiny handles this by providing reactive functions, which come in three main flavours: reactive, render and observe, all of which were used in the application.

3.3.1

Reactive Expressions

Figure 3.10: Example reactive expression [2]



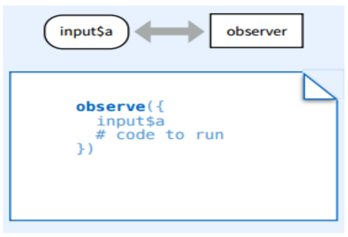
```
x <- reactive({
  input$a
})

output$y <- renderText({
  x()
})
output$z <- renderText({
  x()
})
```

The purpose of reactive expressions is to create objects that will be used in multiple outputs. Reactive expressions will saves its result the first time you run it. In subsequent calls, the reactive expression checks if the saved value needs to be updated (i.e., whether the inputs it depends on have changed). If the value does need to be updated, the reactive object will recalculate it and then cache the new result. If the value is already updated, the reactive expression will return the cached value without running the computations involved in the function[2].
In my application, the Preprocess reactive expression took in data provided by the user and cleaned it so that it was in a suitable format for use throughout the rest of the server. Since this calculation took a few seconds to run in the application, using the cached value greatly sped up interaction times.

```
Preprocess=reactive({
  #do all preprocessing, validation and error checking here and then
  #return processed data frame
   .....
   .....
   ProcessedData
})
```

Observe Expressions

Figure 3.11: Example observe expression [2]



```
observe({
  input$a
  # code to run
})
```

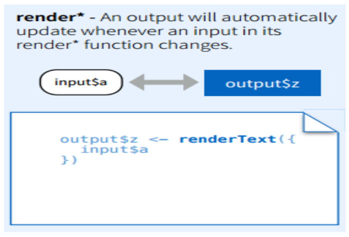
Observe expressions create code that executes when an input changes, but does not directly create an output object. Observers are most often used for their side effects[2]. For example, in my application I wanted the months dropdown to only contain months that athletes trained in for the selected year. An observe expression was used to create a dependency on the barplotyear input. When the barplotyear input changed, the observer ran and updated the barplotMonth input, without returning anything.

```
upDateBarplotMonths=observe({
  #Create dependency on barplotyear – when this changes, dates need to
  #update
  input$barplotYear
  validate(need(exists('ProcessedData'),""))
  updateSelectInput(session, "barplotMonth",
  choices = sort(unique(format(as.Date(ProcessedData[format(as.Date(
  ProcessedData$Date,format="%Y/%m/%d"),'%Y')==input$barplotYear,'Date'],
   format="%Y/%m/%d"),"%m")))
  )
})
```

Render Expressions

Render functions are used in shiny apps to pass plots, charts, text and other output created in the server to the UI to be displayed. An output will automatically update whenever an input in its render function changes. Render expressions typically hold the code that the server needs to rebuild a UI component afer a widget changes[2]. Shiny follows the convention of naming render expressions as render[Output Type], where the outputs range from tables to plots. Render expressions must return a specific type of object to be displayed or else errors are incurred. In my application, the render expressions tended to be lengthy since most of the plots are quite detailed, however a simple example exists in the from of the player summary.

Figure 3.12: Example observe expression [2]



render* - An output will automatically update whenever an input in its render* function changes.

```
output$z <- renderText({
  input$a
})
```

```
output$PlayerSummaryPlot=renderTable({
  input$getname
  playerSummary(input$getname)
})
```

Dependency Graph

Shiny has many debugging tools and tricks that can help the application designer to ensure that their app is running as expected. One such tool is the reactive log visualization. This is essentially a dependency chart that shows how the outputs in an application depend on the inputs. There are three types of nodes in this chart displayed in figures 3.3-3.5. Figure 3.3 denotes nodes that can be thought of as 'endpoints'. Shown below is a dependency graph created before the app was completed. Notice that some nodes have many connections which denotes indicates their importance in the applications. Circled are nodes that have no connections which shows that they are redundant in the application. The nodes shaped in f (show example of 3 dependency in my application)

Figure 3.13: Render & Observe Nodes (Endpoint)

Figure 3.14: Reactive Nodes (Intermediary)

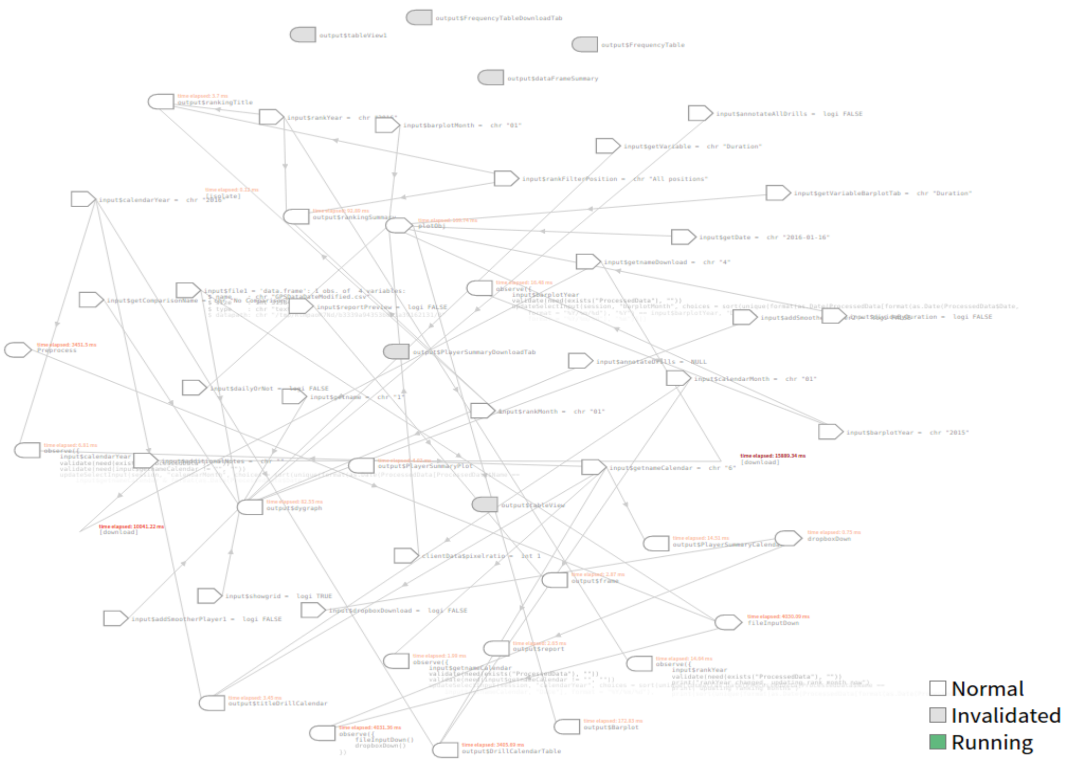
Figure 3.15: Input Nodes (Start Point)

3.4 Program Workflow

The shiny application executes according to the following sequence:

1. UI file runs everything above the shinyUI function once (show image in parallel)

Figure 3.16: Reactive Log Graph



2. UI file runs the code inside the shinyUI function once

3. Server file runs the code outside the shinyServer function once

4. Server file runs any code outside render, reactive or observe functions once

5. Server file runs code inside reactive expressions multiple times according to the dependency list that Shiny has built up. Whenever an input in a reactive expression changes, the expression will re-run.

3.5 User Interface

3.6 Testing

As the app grew in complexity, it grew frustrating to change a piece of functionality and then manually run the app and check each tab to ensure that the app still operated as intended. I wrote a short set of unit tests in the python language using the selenium package in order to automate this process. This meant that every time I made a significant change to the app, I could run the tests while working on some other part of the application without having to manually ensure everything was present. I wrote the tests in python because the application needed to be running in order to use the selenium webdriver to navigate it and because python is more suited to testing web applications than R. Code can be found for the tests on github at _____.

3.7 Conclusion

Conclusion

Bibliography

[1] GPSports FAQ
`http://www.gpsports.com/support/FAQ_GPS_Tracking.pdf`

[2] Shiny Cheat Sheet
`https://www.rstudio.com/wp-content/uploads/2015/02/shiny-cheatsheet.pdf`

[3] R for Data Science
`http://r4ds.had.co.nz/exploratory-data-analysis.html`

[4] GPSports variables analysis
`http://gpsports.com/gpsports-101/`

[5] GPSports New.Bodyload documentation
`http://www.gpsports.com/support/UnderstandingBodyLoad.pdf`

[6] An Introduction to Statistical Learning with Applications in R
`http://www-bcf.usc.edu/ gareth/ISL/ISLR%20Sixth%20Printing.pdf`

[7] The Elements of Statistical Learning: Data Mining, Inference, and Prediction.
`http://statweb.stanford.edu/ tibs/ElemStatLearn/`

[8] Detecting Multicollinearity Using Variance Inflation Factors
`https://onlinecourses.science.psu.edu/stat501/node/347`

[9] Multicollinearity in Regression Models
`http://sites.stat.psu.edu/ ajw13/SpecialTopics/multicollinearity.pdf`