Markdown Final Project

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
install.packages('tidyverse')
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
dat <- read.csv('UFCData.csv')
library(ggplot2)
```

```
library(MASS)
#install.packages('GGally')
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
install.packages('data.table')
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library(data.table)
```

```
install.packages('corrplot')
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Data Exploration

In this section, we will mutate and plot the data in order to understand relationships within the data.

```
#head(dat)
nrow(dat)
```

```
## [1] 3592
```

Column mutation

converting Blue and red on winner into to Win /Loss /Draw /No Contest for each fighter

```r
#convert Blue and red on winner into  to Win /Loss /Draw /No Contest for each fighter
 result1<-rep("a", 3592)
result2<-rep("a", 3592)
for (i in c(1:3592)) {
  if (dat$Winner[i]=="Red") {
    result1[i]<-"L"
    result2[i]<-"W"
  } else if (dat$Winner[i]=="Blue") {
   result1[i]<-"W"
    result2[i]<-"L"
  } else if (dat$Winner[i]=="draw") {
    result1[i]<-"D"
    result2[i]<-"D"
  } else {
    result1[i]<-"NC"
    result2[i]<-"NC"
   }
 }
```

```r
dat$Blue_result <- result1
dat$Red_result <- result2
```

```r
table(dat$Winner)
```

```
##
## Blue  Red
## 1212 2380
```

```r
table(dat$no_of_rounds)
```

```
##
##    1    2    3    4    5
##   27   33 3135    1  396
```

```r
table(dat$R_age)
```

```
##
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
##   3  13  22  46 101 154 175 258 288 304 339 345 317 294 221 217 152 121  83  57
##  39  40  41  42  43  44  45  46
##  30  28   8   5   3   5   1   2
```

```r
table(dat$B_age)
```

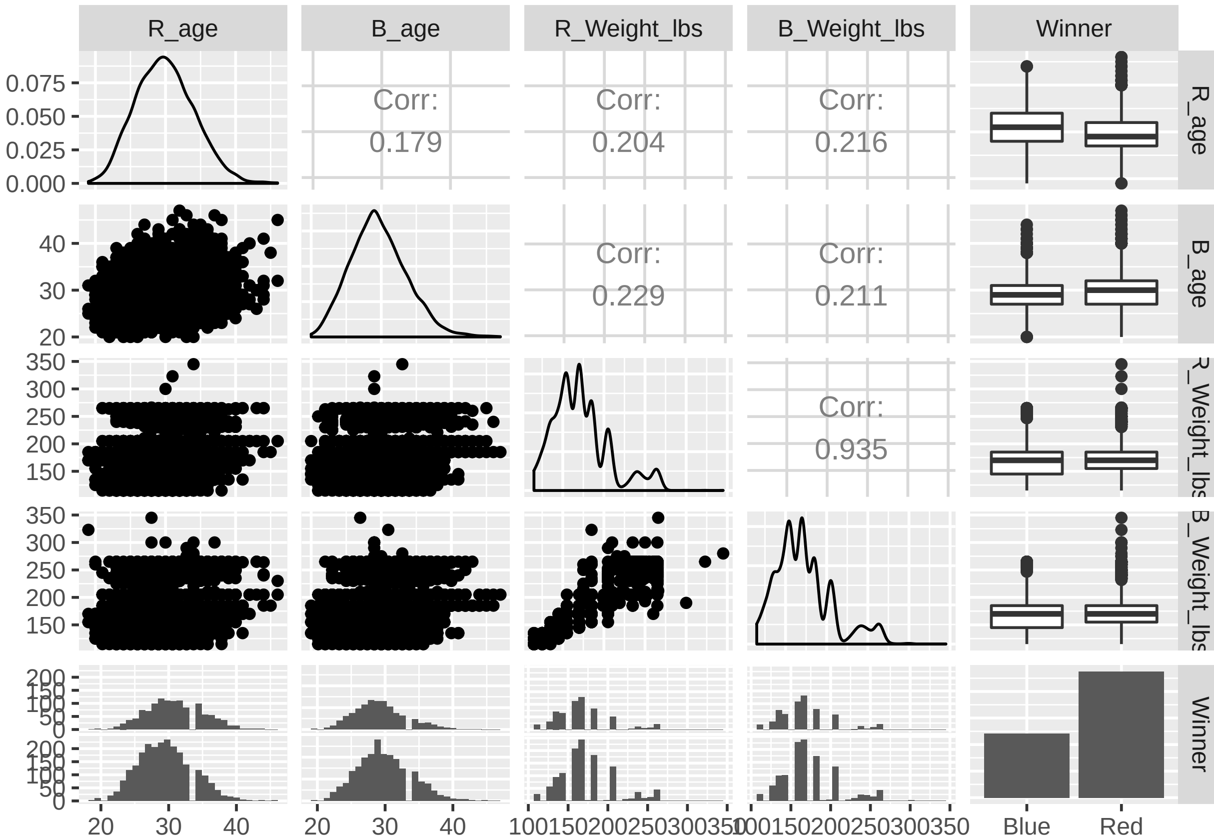
```
##
##  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
##   7  20  58 103 141 212 252 312 347 411 343 315 265 210 182 118 111  69  38  28
##  40  41  42  43  44  45  46  47
##  15  11  10   5   3   3   2   1
```

First off, let's get a broad look at how several of the columns within this dataset look in relationship to each other.

```r
ggpairs(data=dat, columns=c("R_age", "B_age", "R_Weight_lbs", "B_Weight_lbs", "Winner"))
```

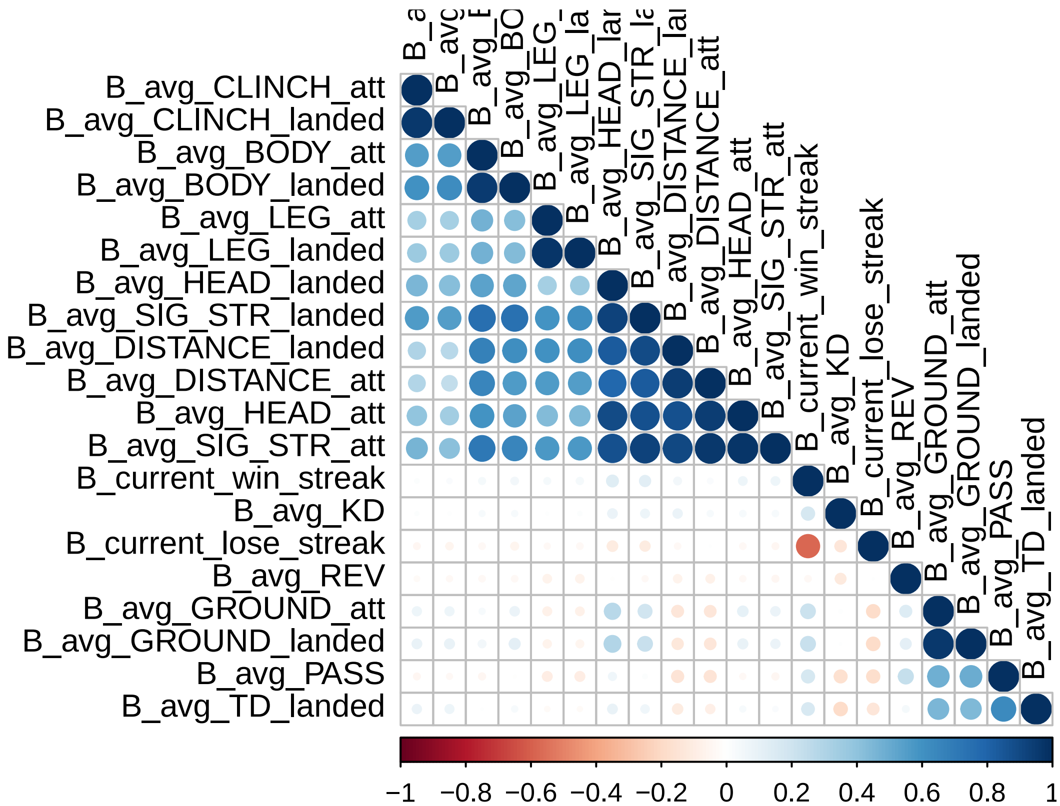
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



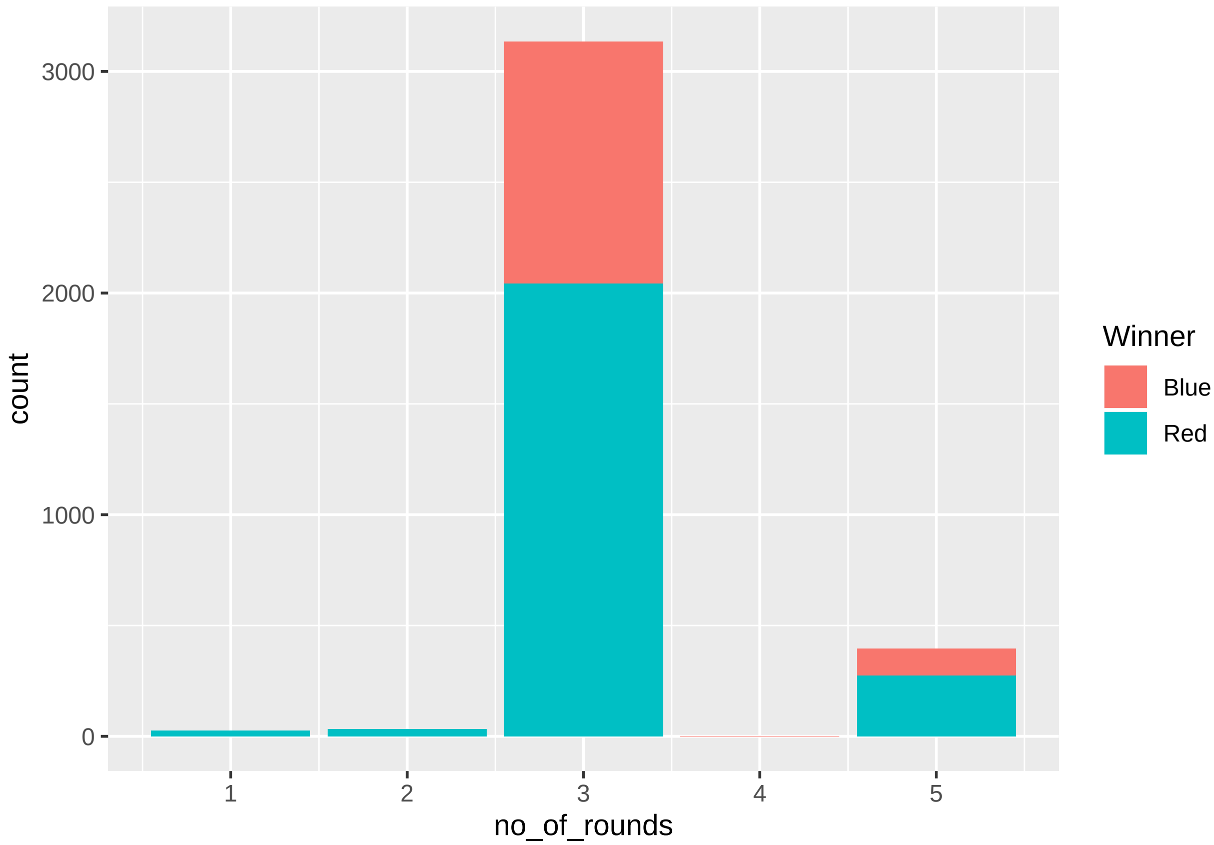
Testing for corelation To see if some of the variables are correlated to each other we run the corelation plots, which produced the results below:

```
totalcor<-cor(dat[,c(4:5,7:23,27)])
par(xpd=TRUE)
corrplot(totalcor, type = "lower", order = "hclust",
         tl.col = "black", tl.srt = 90, mar = c(1,1,.5,.5))
```

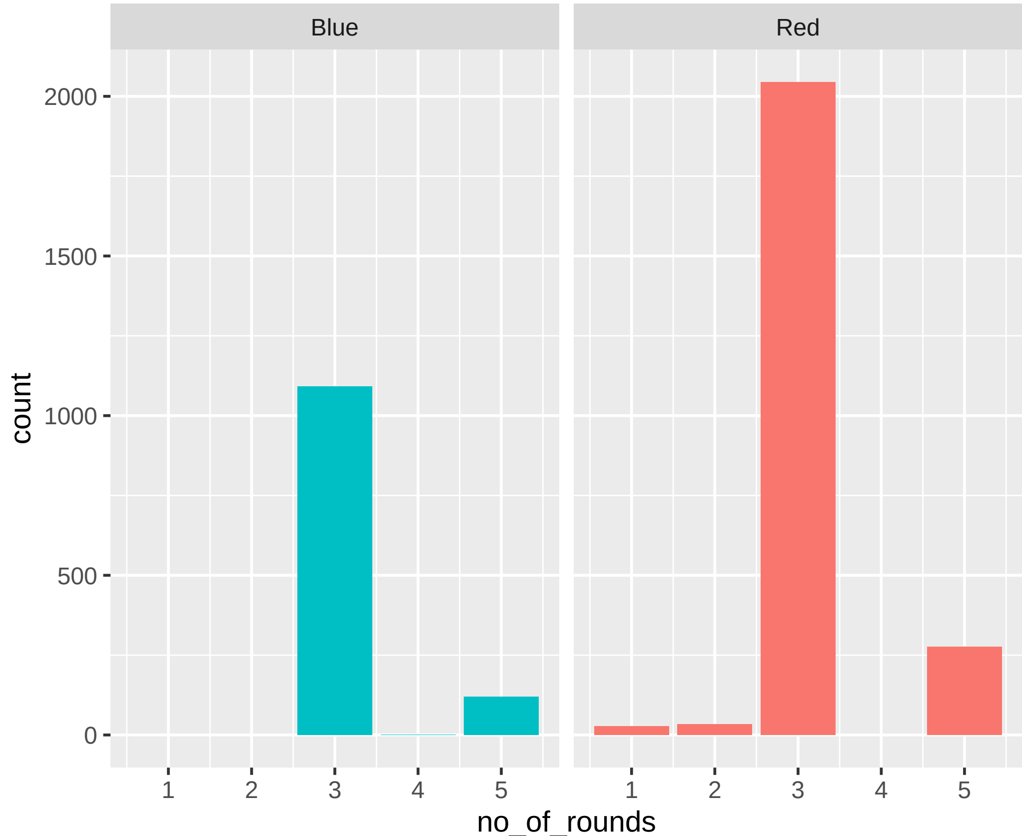


##Testing whether the variables are normally distributed The variables in our data are not normally distributed.

```
ggplot(data=dat, aes(no_of_rounds)) + geom_bar(aes(fill=Winner))
```

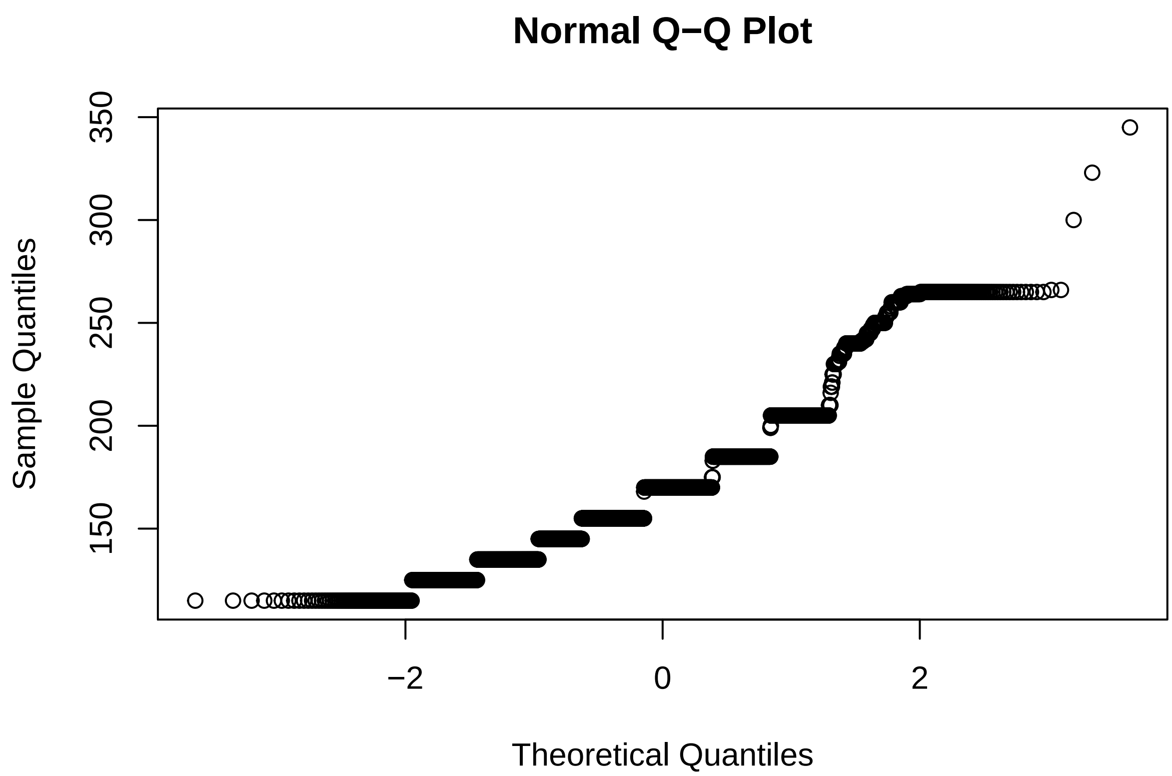


Now, in order to get a feel for how the wins are distributed, we can use a facet grid to see wins for each color by

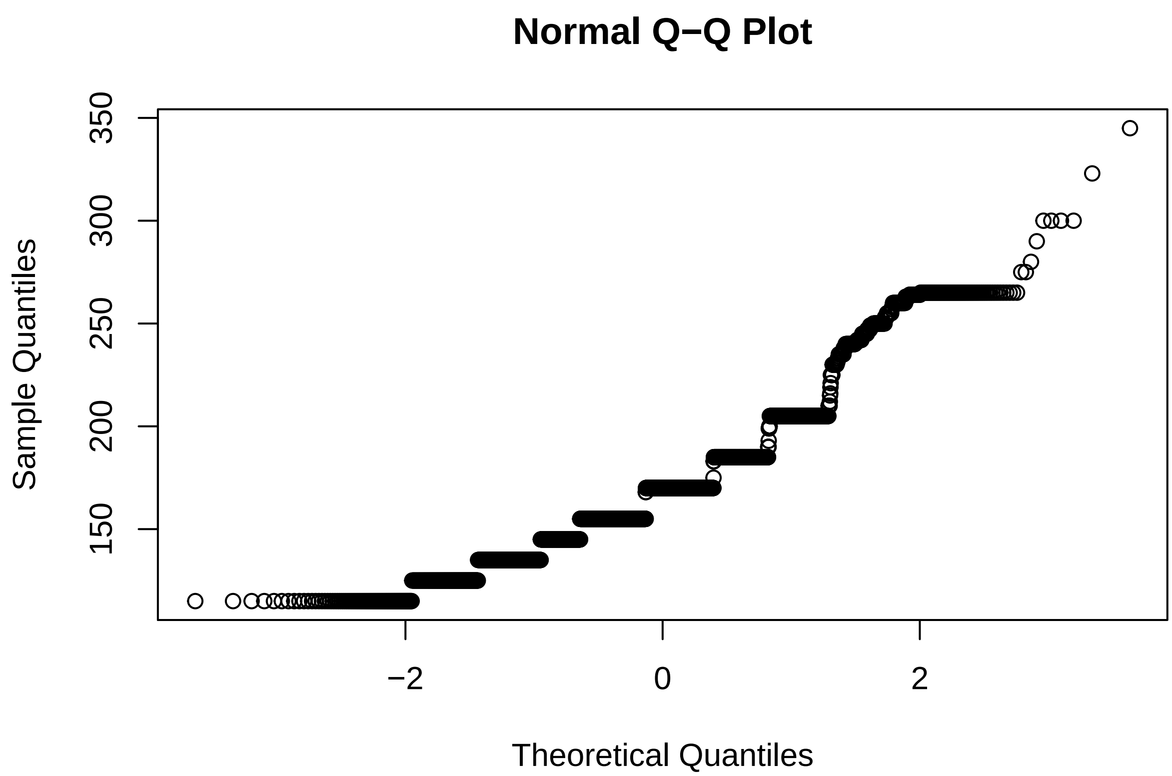


the number of rounds each fight took.
So this plot shows the number of times blue and red each one, with the number of rounds that fight went on the x-axis. The first graph is the number of times Blue won, with most occurring in the third round, and the second shows the number of times red won, again, with the majority ocurring in the third round.

```
dat$weight_dif <- dat$R_Weight_lbs - dat$B_Weight_lbs
dat$height_dif <- dat$R_Height_cms - dat$B_Height_cms
qqnorm(dat$R_Weight_lbs)
```



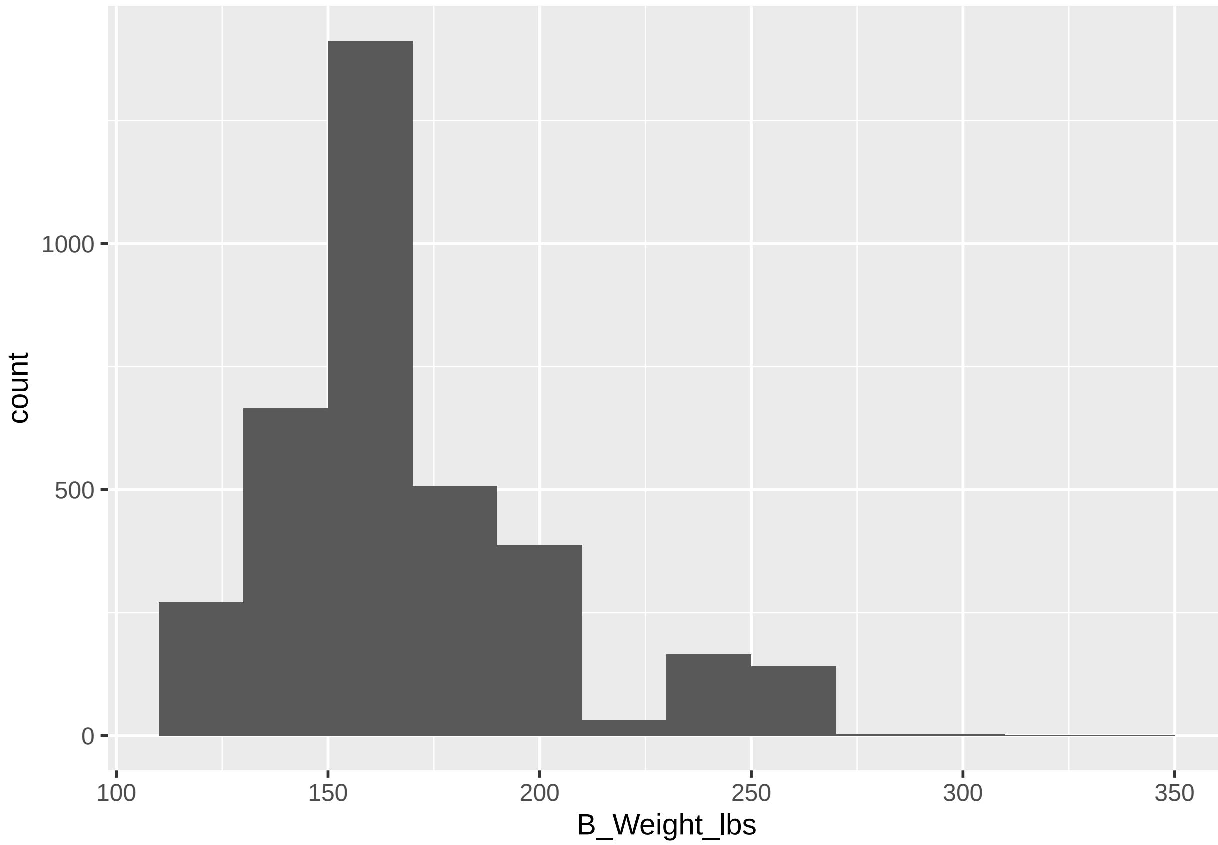

```
qqnorm(dat$B_Weight_lbs)
```



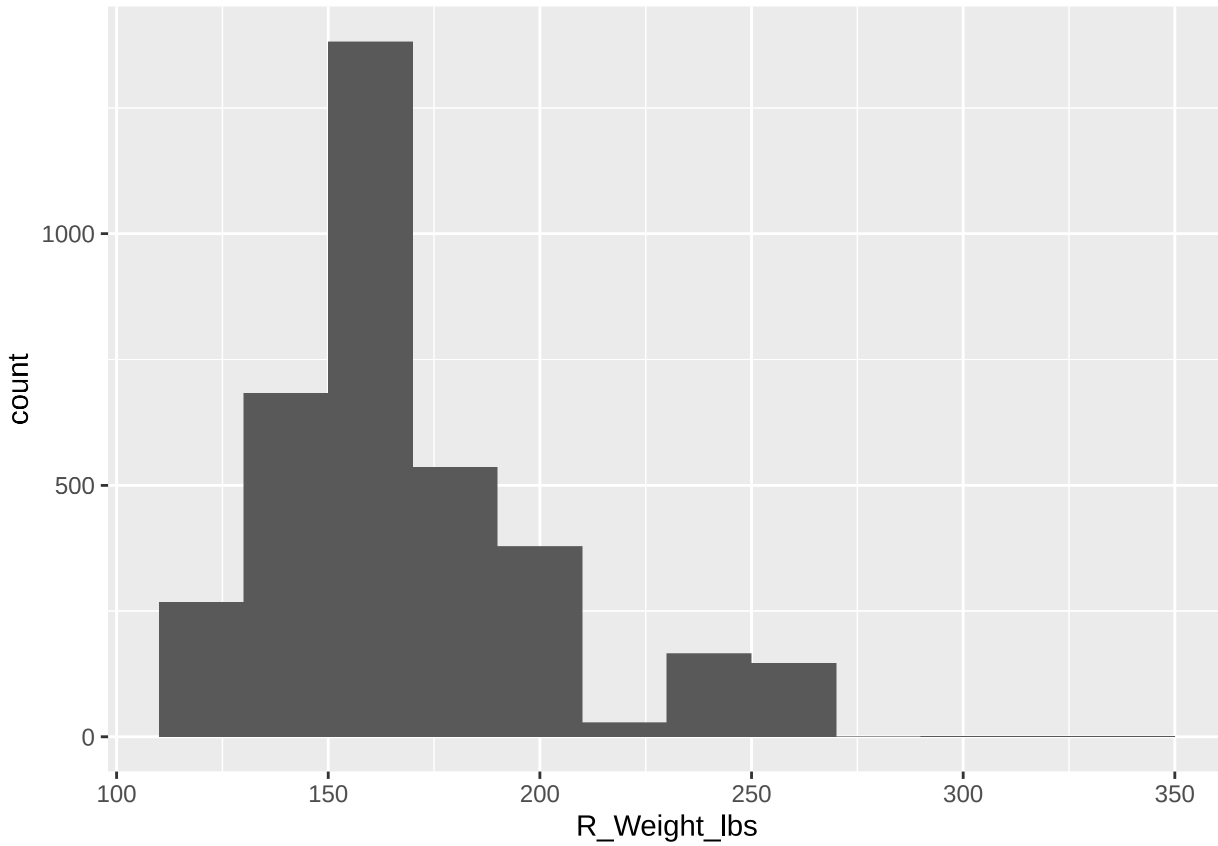
So we see that most of our fighters' weight is distributed within 2 standard deviations of the mean, which appears to be around 190 pounds.

To examine this further, we plotted histograms of both red and blue fighters' weights.

```r
ggplot(data=dat, aes(B_Weight_lbs)) + geom_histogram(binwidth = 20)
```

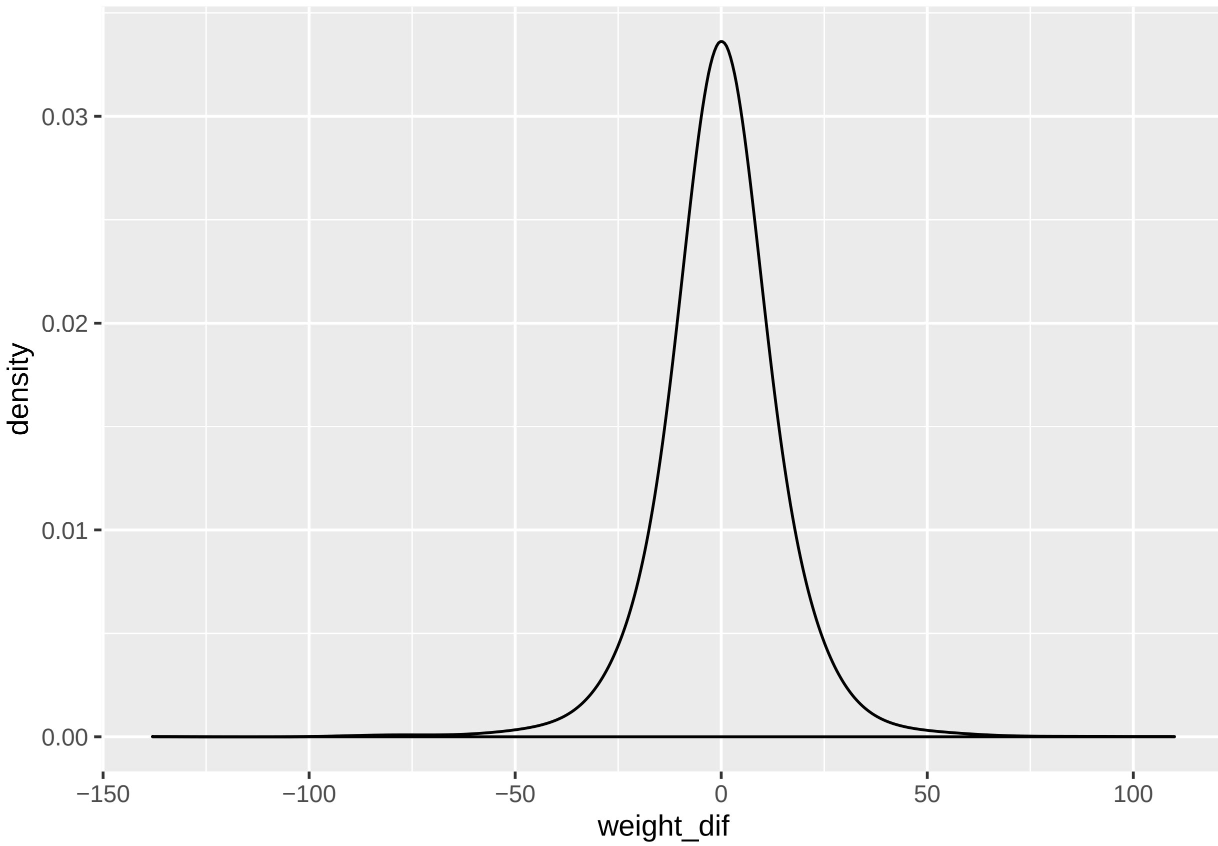



```r
ggplot(data=dat, aes(R_Weight_lbs)) + geom_histogram(binwidth = 20)
```



Now, let's see how the weight difference in each fight is distrubuted.

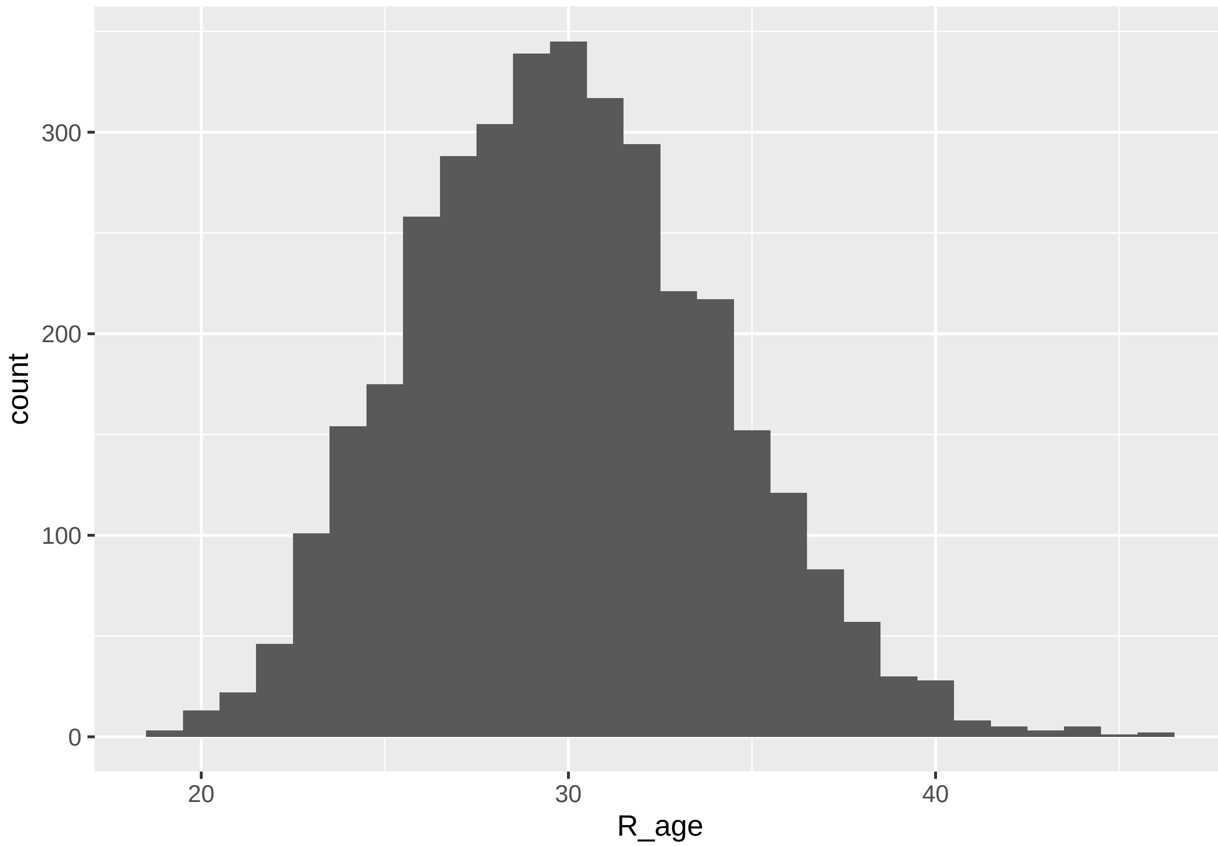
```
ggplot(data=dat, aes(x=weight_dif)) + geom_density(adjust=4)
```



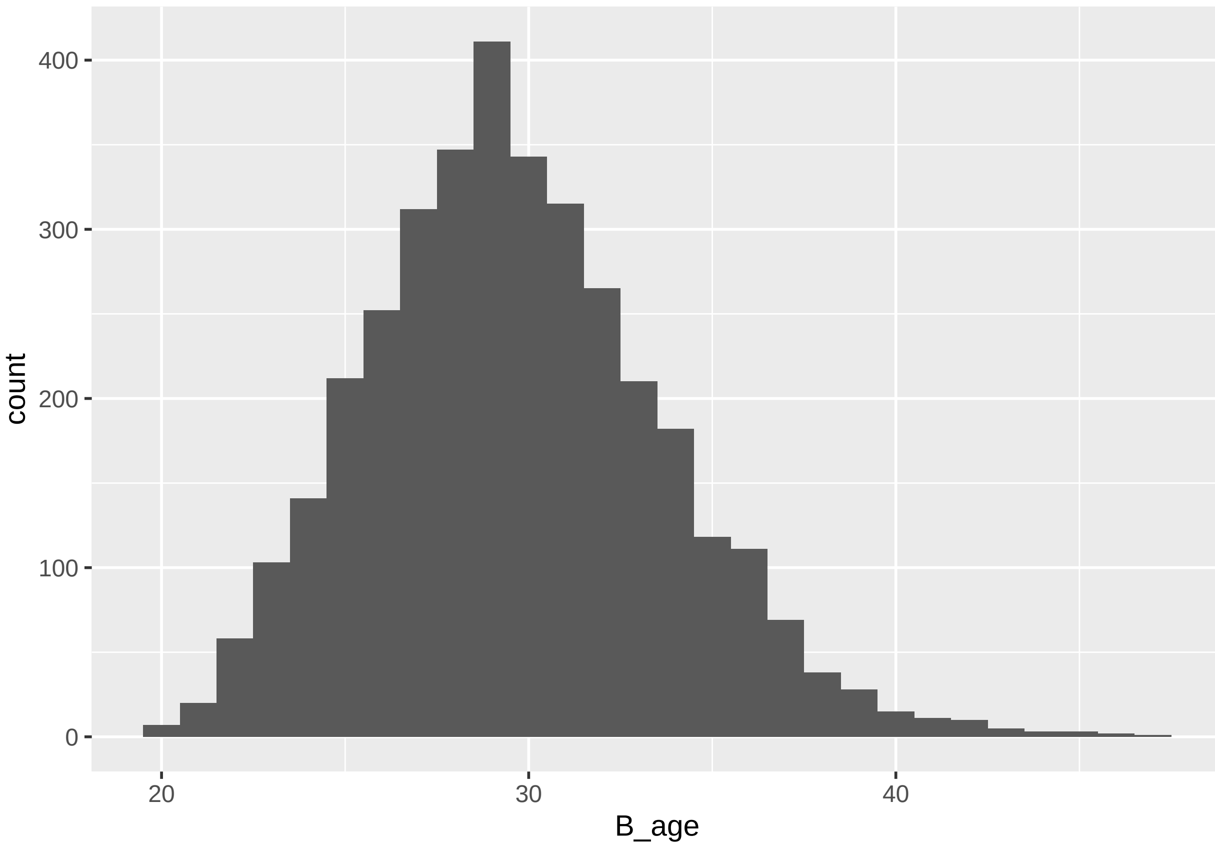
And as to be expected, since every fight has its own weight class, the difference in weights will be pretty minimal, centered around zero.

Moving on, lets take a look at age for both Red and Blue fighters

```
ggplot(data=dat, aes(R_age)) + geom_histogram(binwidth=1)
```




```
ggplot(data=dat, aes(B_age)) + geom_histogram(binwidth=1)
```



In both plots, around 28-30 is the most ocurring age with it quickly tapering off after 34 years old, which makes intuitive sense. Fighting is very tough on the body, and one can only take so much physical abuse before having to retire.

Linear Regression Analyis

Null Hypothesis: No difference in the number of head strikes attempted by red and blue between winners.

Building a Linear Regression Model

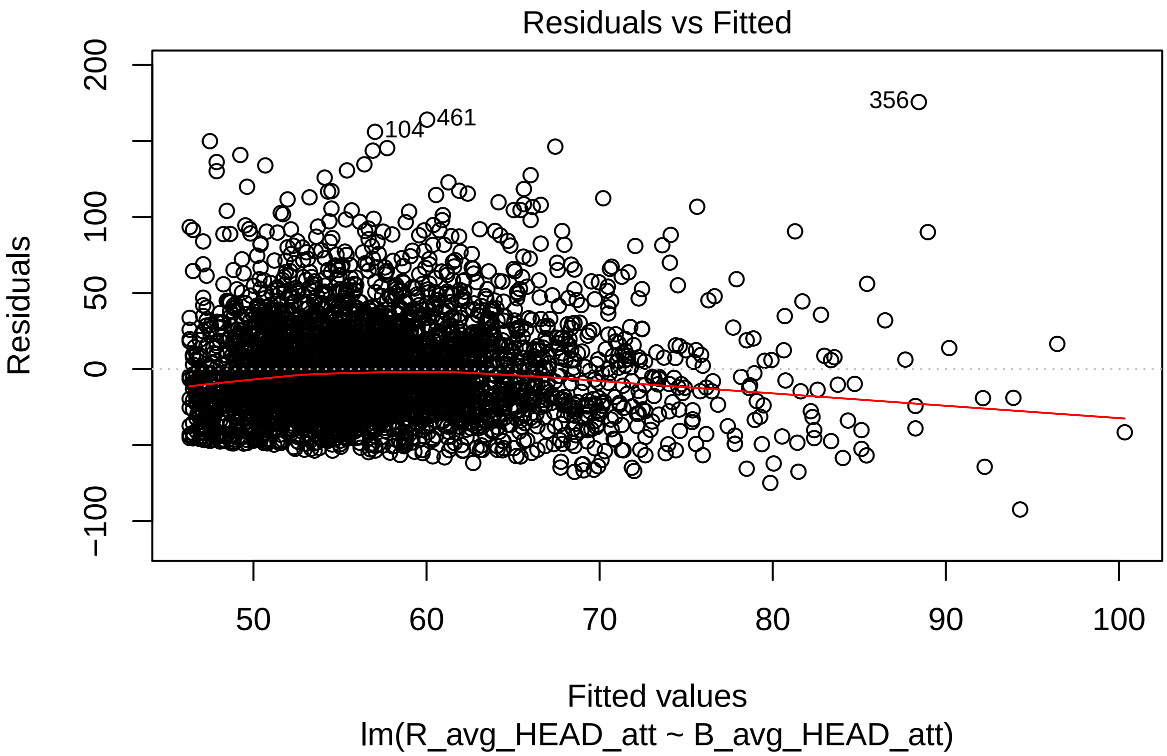
```r
fit<-lm(R_avg_HEAD_att ~ B_avg_HEAD_att, data = dat)
summary(fit)
```

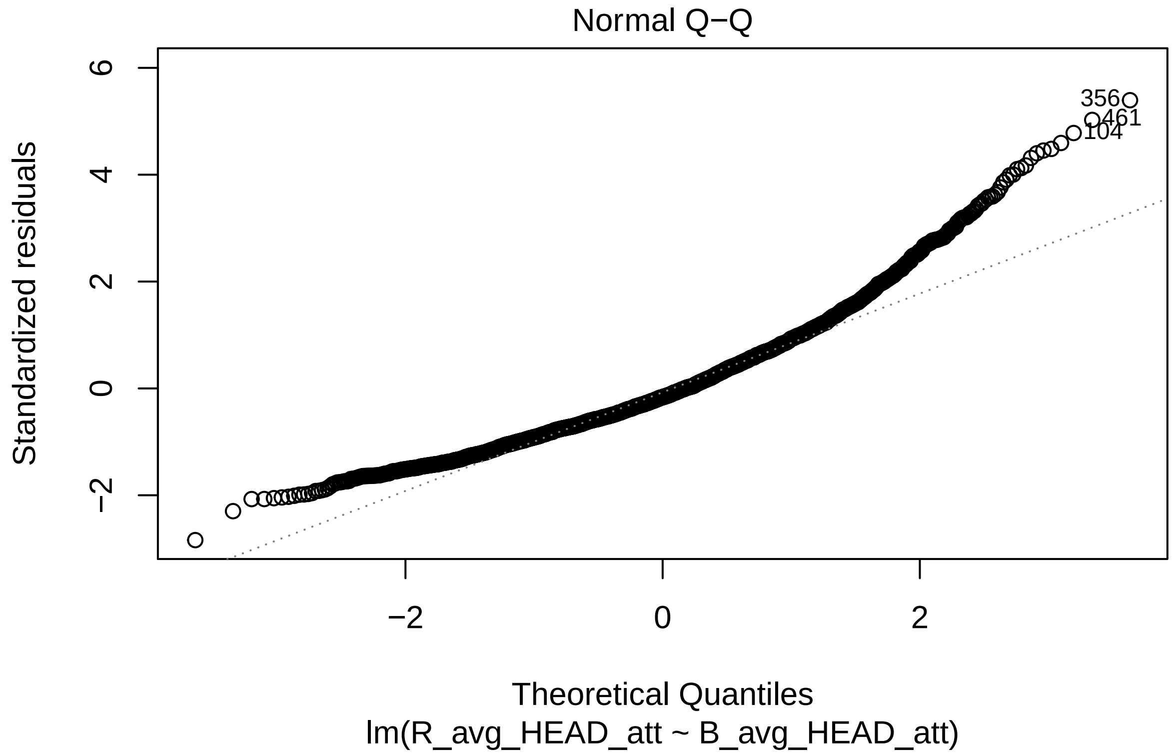
```
##
## Call:
## lm(formula = R_avg_HEAD_att ~ B_avg_HEAD_att, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -92.290 -22.680  -5.325  17.966 175.560
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     46.31550    1.00561   46.06   <2e-16 ***
## B_avg_HEAD_att  0.19502    0.01505   12.96   <2e-16 ***
## ---
```

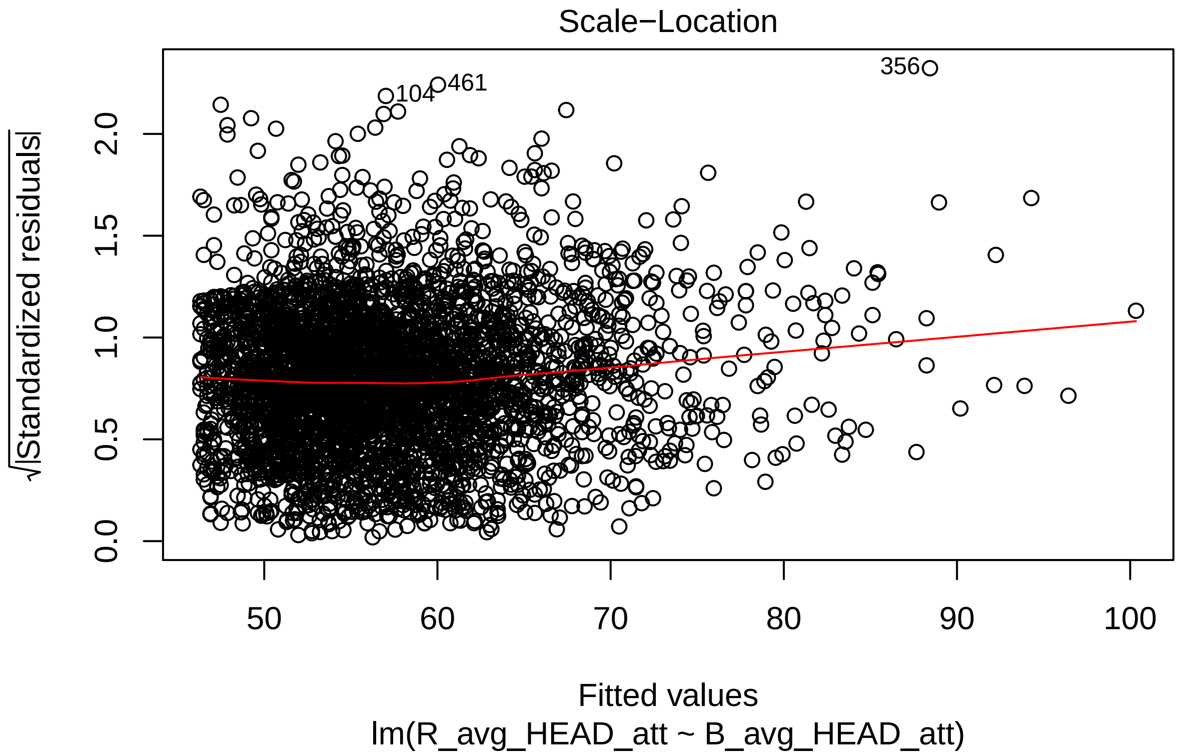
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.63 on 3590 degrees of freedom
## Multiple R-squared:  0.04468,    Adjusted R-squared:  0.04441
## F-statistic: 167.9 on 1 and 3590 DF,  p-value: < 2.2e-16
```

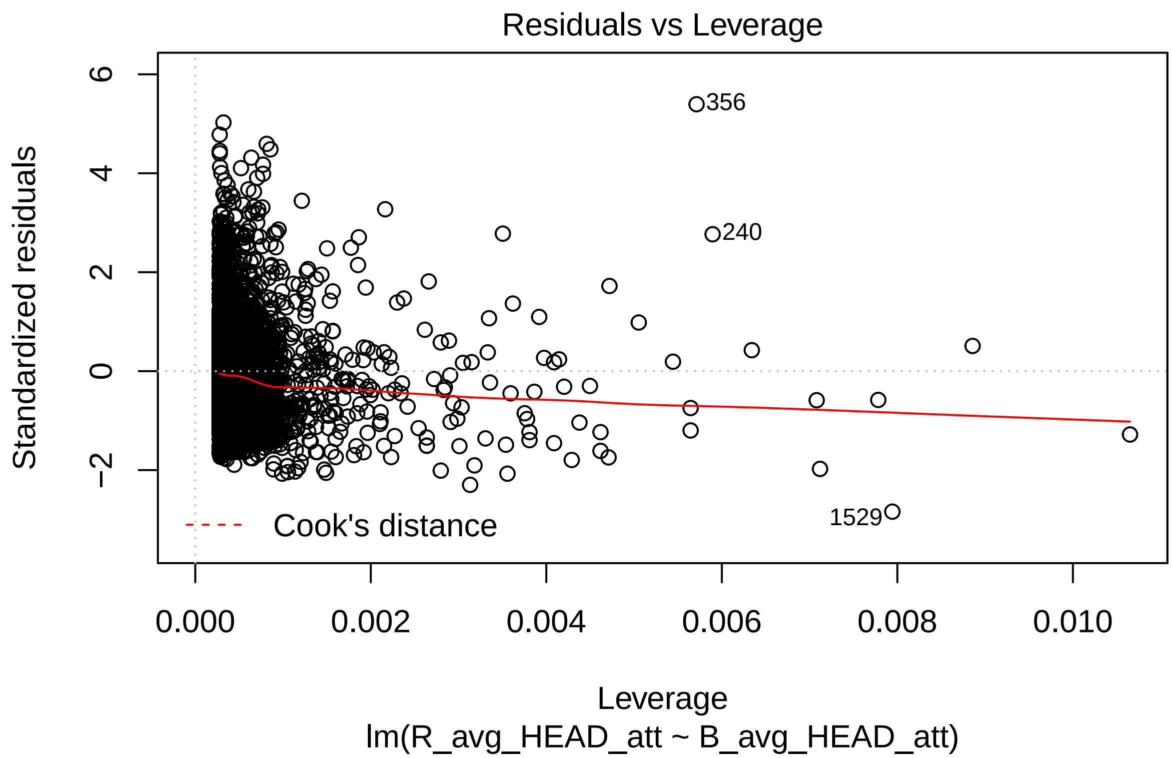
Plotting the regression

```r
plot(lm(R_avg_HEAD_att ~ B_avg_HEAD_att, data = dat))
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(R_avg_HEAD_att ~ B_avg_HEAD_att)
356
461
104
−2
0
2
4
6


Scale−Location
√|Standardized residuals|
Fitted values
lm(R_avg_HEAD_att ~ B_avg_HEAD_att)
356
104
461
0.0
0.5
1.0
1.5
2.0
50
60
70
80
90
100



Looking at the plots above we can see that the data is not normally distributed, there is not a mean of zero, and there is uncommon variance.

Building a Linear Regression Model

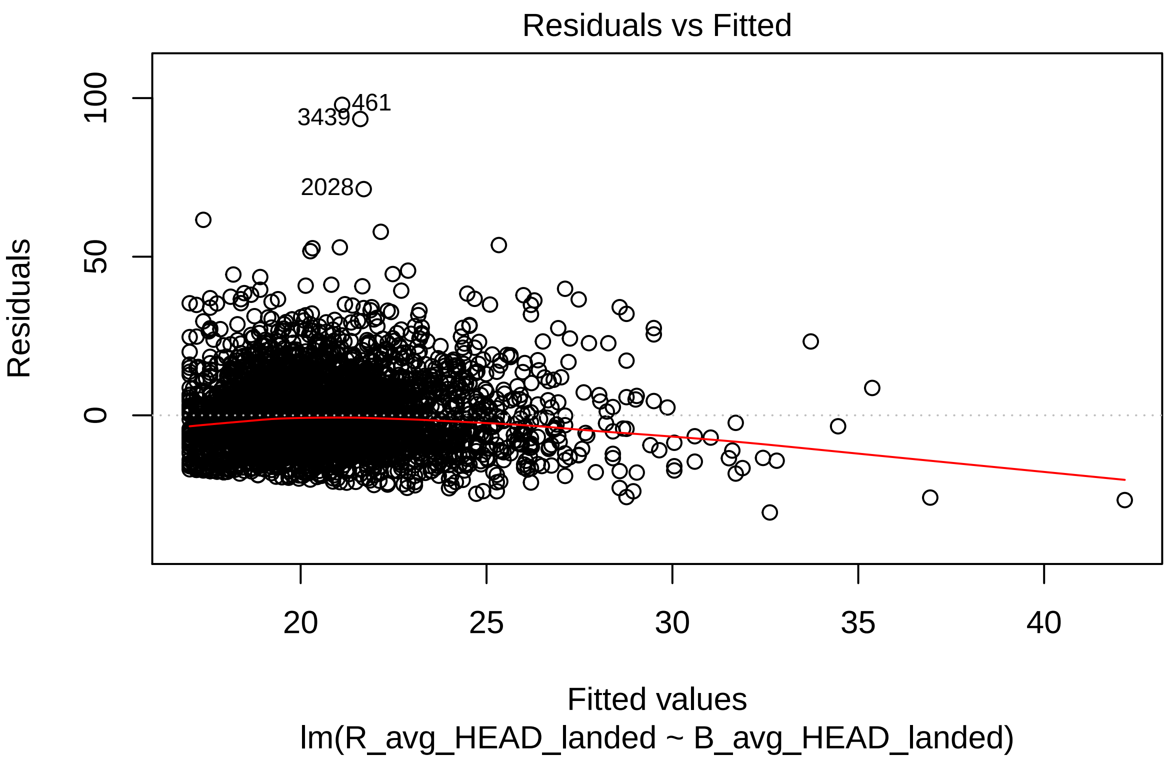
Null Hypothesis: No difference in the number of head strikes landed by red and blue between winners.

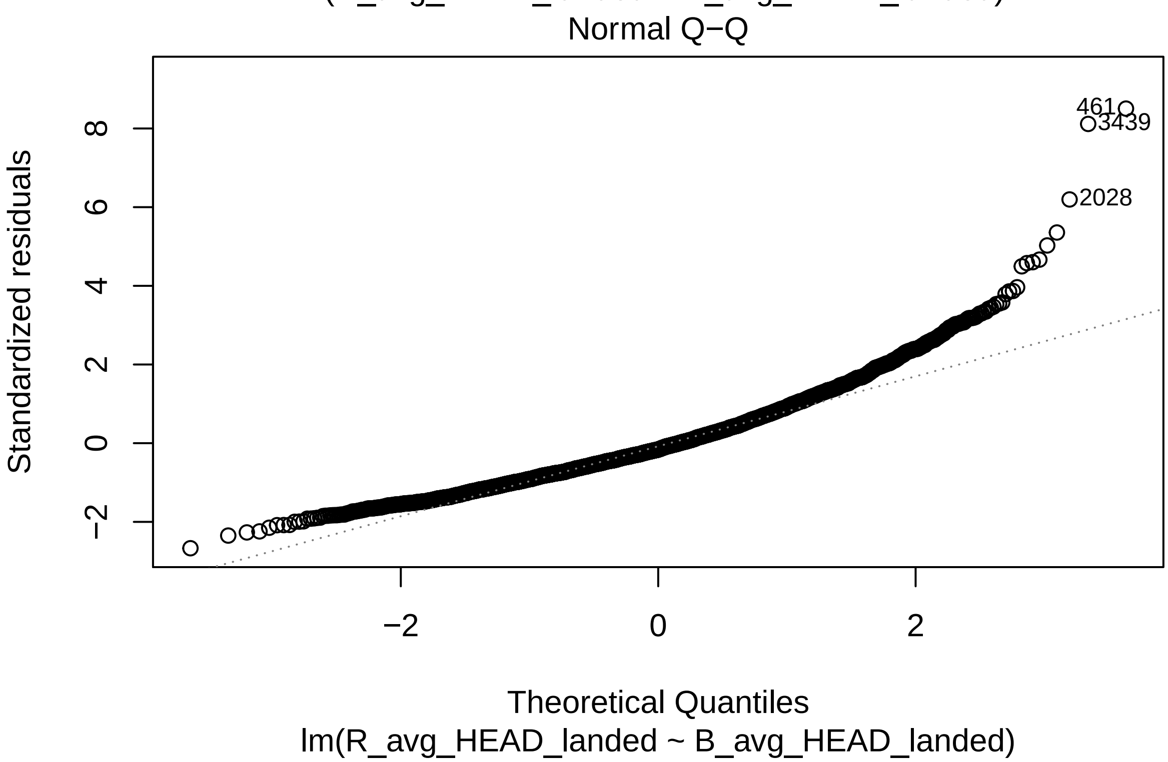
```
fit<-lm(R_avg_HEAD_landed ~ B_avg_HEAD_landed, data = dat)
summary(fit)
```

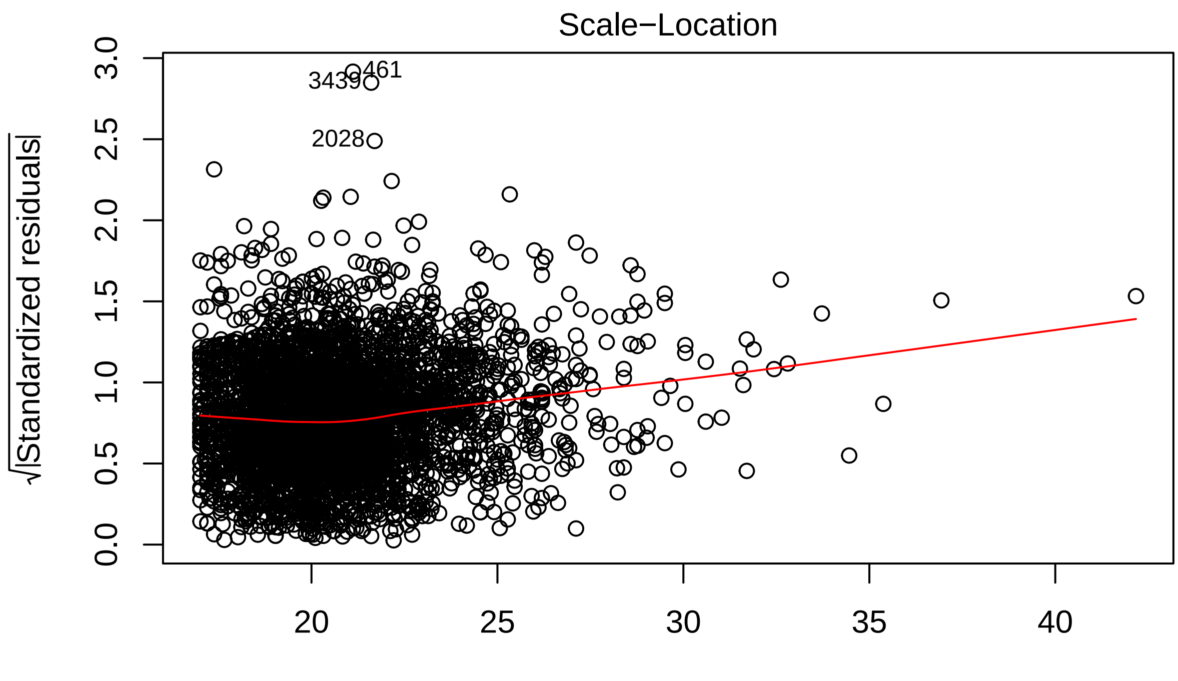
```
##
## Call:
## lm(formula = R_avg_HEAD_landed ~ B_avg_HEAD_landed, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.622  -7.805  -1.849   5.995  97.885
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       17.01427    0.35328   48.16   <2e-16 ***
## B_avg_HEAD_landed  0.18362    0.01473   12.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.51 on 3590 degrees of freedom
## Multiple R-squared:  0.04146,    Adjusted R-squared:  0.0412
## F-statistic: 155.3 on 1 and 3590 DF,  p-value: < 2.2e-16
```

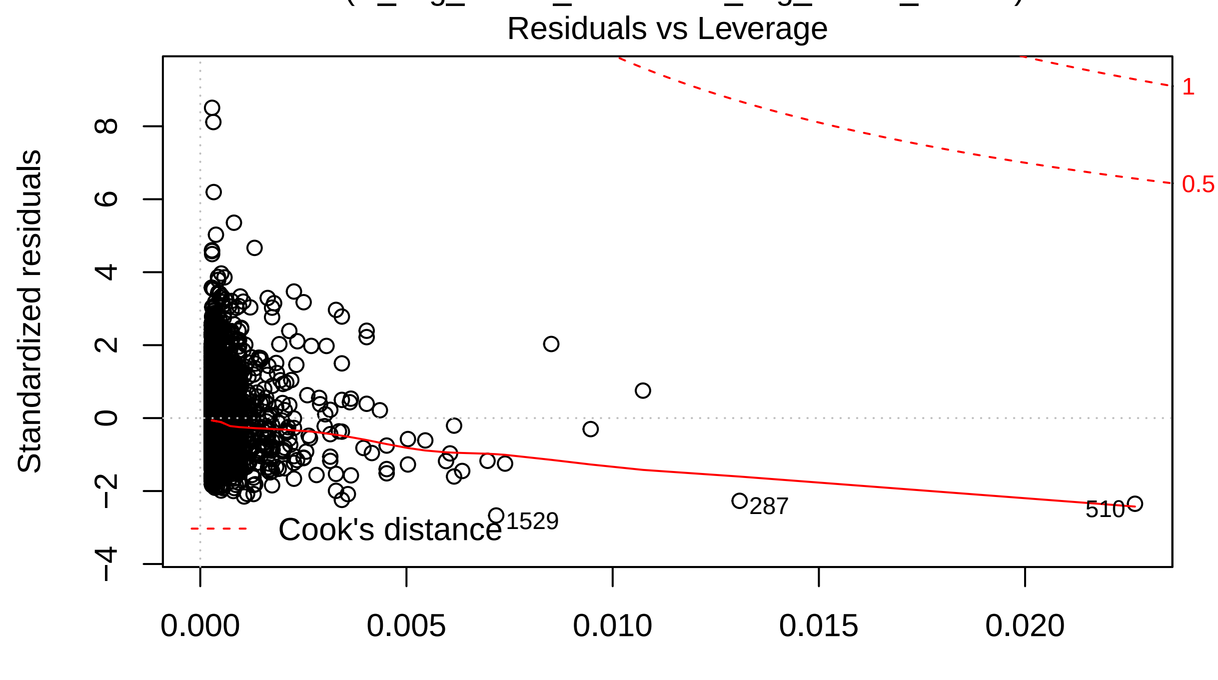
Plotting the regression model

```
plot(lm(R_avg_HEAD_landed ~ B_avg_HEAD_landed, data = dat))
```









Looking at the plots above we can see that the data is not normally distributed, and there is uncommon variance.

Logistic Regresion

So now that we have been able to visualize various parts of the data, we can start creating models that might describe the relationships within our data set.

First we need to create a category for our winner: 1 if red, 0 if blue

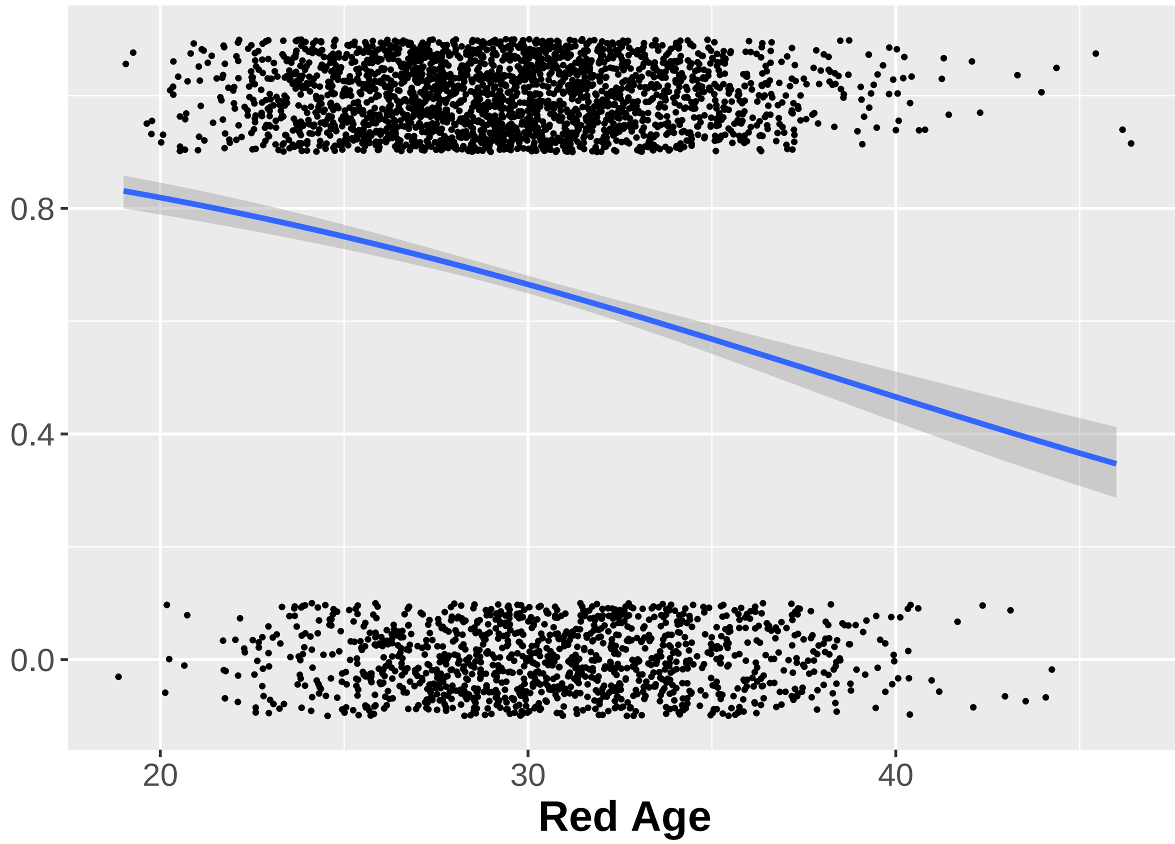
```
dat$numeric_winner <- as.numeric(dat$Winner == "Red")
```

And now we will create a logistic regression throwing a bunch of explanatory variables:

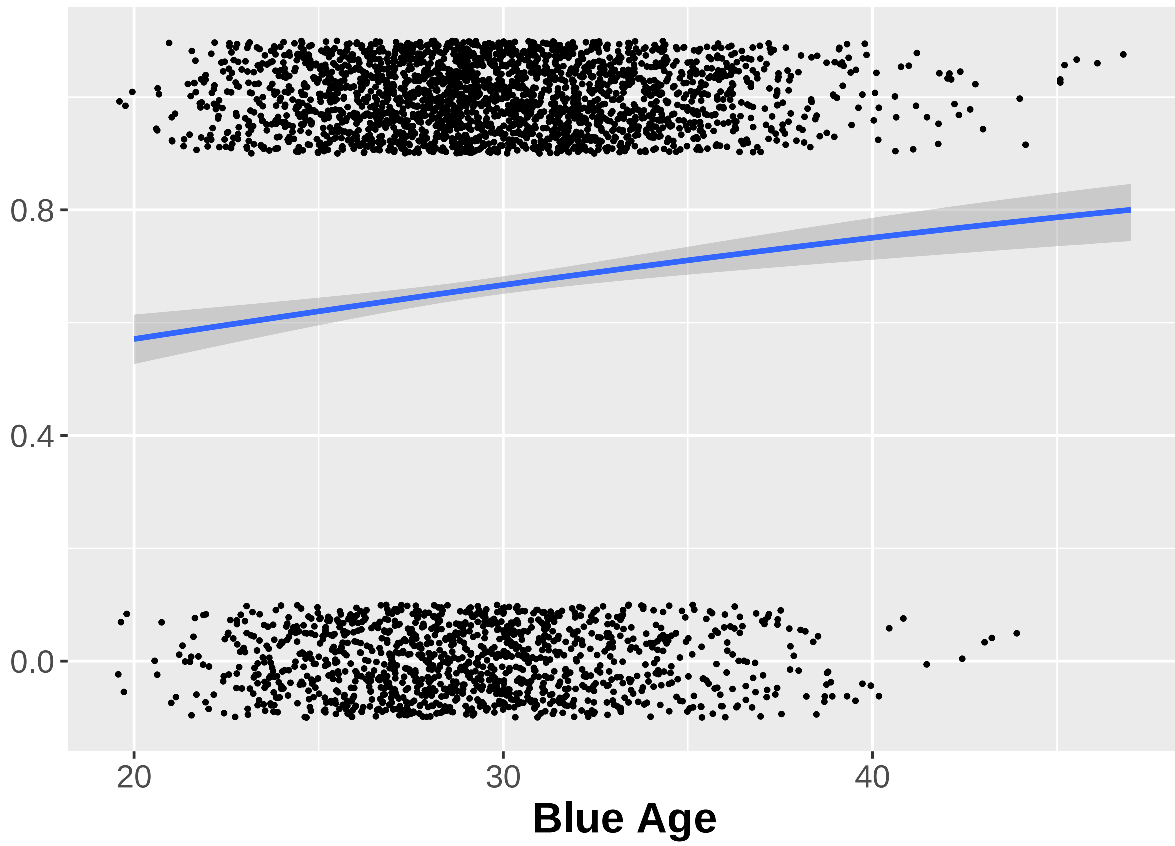
```
logit1 <- glm(numeric_winner~R_Height_cms + B_Height_cms + R_age + B_age + R_Weight_lbs + B_Weight_lbs
summary(logit1)
```

```
##
## Call:
## glm(formula = numeric_winner ~ R_Height_cms + B_Height_cms +
##     R_age + B_age + R_Weight_lbs + B_Weight_lbs + weight_dif +
##     height_dif + B_avg_BODY_att + R_avg_BODY_att + B_avg_BODY_landed +
##     R_avg_BODY_landed + B_avg_HEAD_att + R_avg_HEAD_att + B_avg_HEAD_landed +
##     R_avg_HEAD_landed, family = "binomial", data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1853  -1.2261   0.7026   0.9073   1.7523
##
## Coefficients: (2 not defined because of singularities)
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        6.5521530  1.3754957   4.763 1.90e-06 ***
## R_Height_cms      -0.0173155  0.0073434  -2.358   0.0184 *
## B_Height_cms      -0.0107882  0.0074939  -1.440   0.1500
## R_age             -0.0980670  0.0095999 -10.215  < 2e-16 ***
## B_age              0.0504323  0.0098410   5.125 2.98e-07 ***
## R_Weight_lbs       0.0066393  0.0034746   1.911   0.0560 .
## B_Weight_lbs       0.0007758  0.0033824   0.229   0.8186
## weight_dif               NA         NA      NA       NA
## height_dif               NA         NA      NA       NA
## B_avg_BODY_att    -0.0404561  0.0195167  -2.073   0.0382 *
## R_avg_BODY_att    -0.0613298  0.0213668  -2.870   0.0041 **
## B_avg_BODY_landed  0.0277643  0.0259989   1.068   0.2856
## R_avg_BODY_landed  0.0638652  0.0292051   2.187   0.0288 *
## B_avg_HEAD_att    -0.0018184  0.0024806  -0.733   0.4635
## R_avg_HEAD_att    -0.0062070  0.0025176  -2.465   0.0137 *
## B_avg_HEAD_landed -0.0069366  0.0064856  -1.070   0.2848
## R_avg_HEAD_landed  0.0153482  0.0067816   2.263   0.0236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4592.8  on 3591  degrees of freedom
## Residual deviance: 4323.7  on 3577  degrees of freedom
## AIC: 4353.7
##
## Number of Fisher Scoring iterations: 4
```

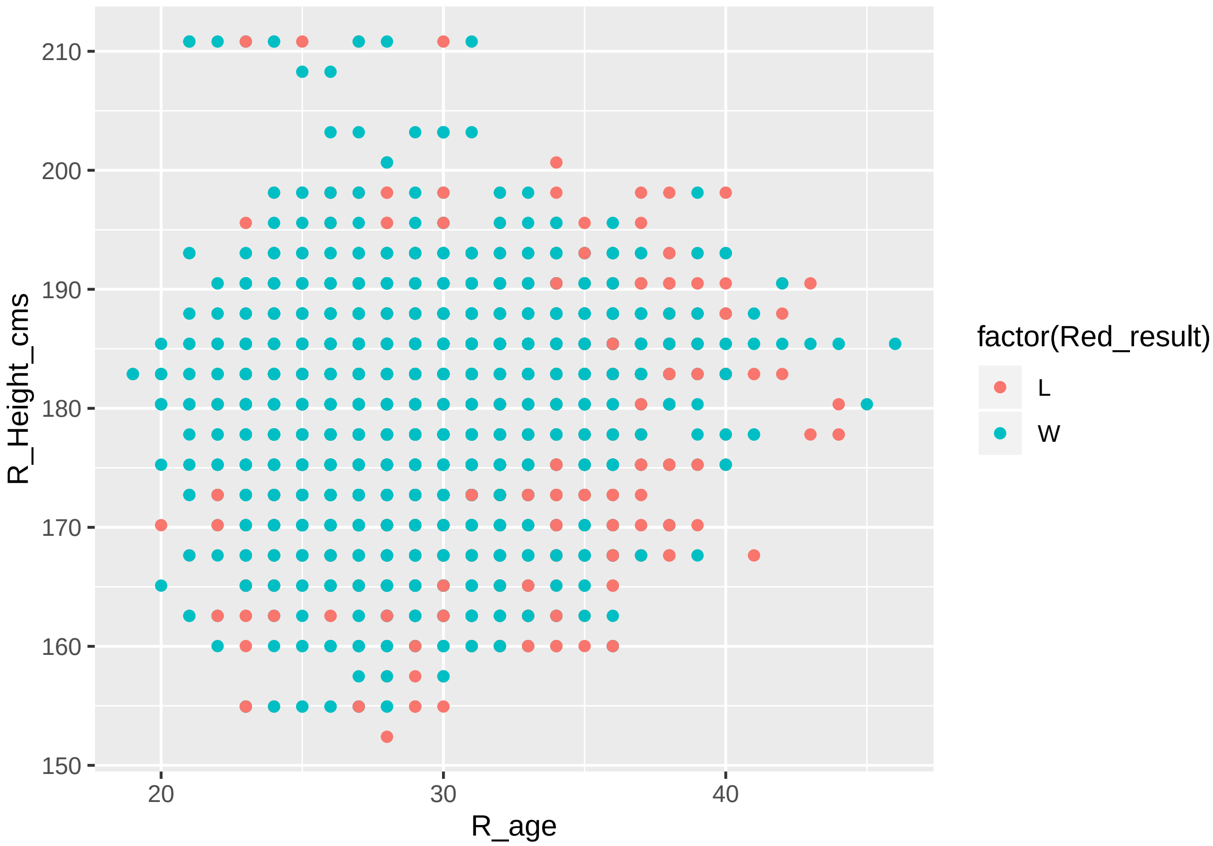
```r
ggplot(dat, aes(x=R_age, y=numeric_winner)) + geom_jitter(height=0.1,width=0.5,size = .5) +
     geom_smooth(method = "glm", method.args = list(family = "binomial")) + xlab("Red Age") +
     ylab(" ") + theme(axis.text=element_text(size=12),axis.title=element_text(size=16,face="bold"))
```




```r
ggplot(dat, aes(x=B_age, y=numeric_winner)) + geom_jitter(height=0.1,width=0.5,size = .5) +
     geom_smooth(method = "glm", method.args = list(family = "binomial")) + xlab("Blue Age") +
     ylab(" ") + theme(axis.text=element_text(size=12),axis.title=element_text(size=16,face="bold"))
```



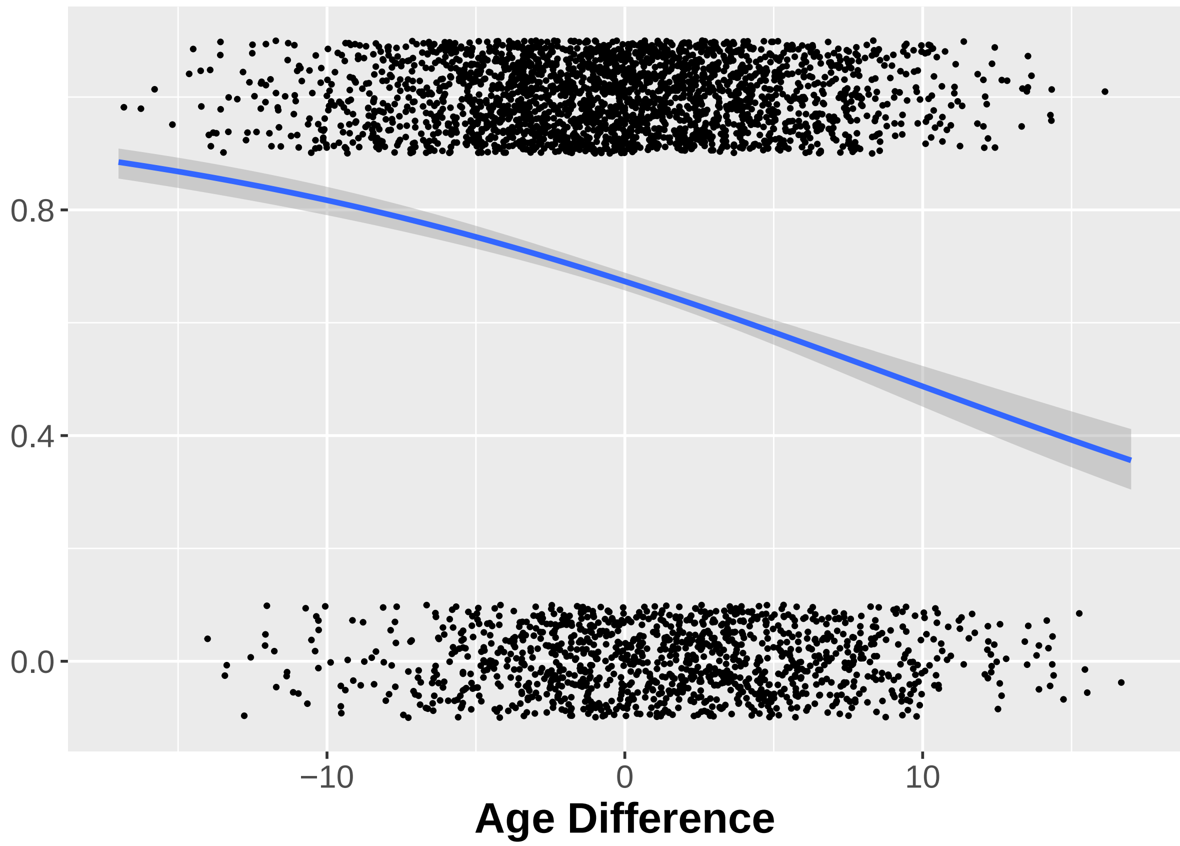

```
ggplot(dat,aes(x=R_age,y=R_Height_cms))+ geom_point(aes(color=factor(Red_result)))
```



So for clarity: the red dots are where the Red fighter lost, and the blue dots are where the Red fighter won

Now perhaps the age of either one is not very interesting to consider, so let's instead take a look at the age difference, specifically the age of red less the age of blue:

```
dat$age_diff <- dat$R_age-dat$B_age
ggplot(dat, aes(x=age_diff, y=numeric_winner)) + geom_jitter(height=0.1,width=0.5,size = .5) +
    geom_smooth(method = "glm", method.args = list(family = "binomial")) + xlab("Age Difference") +
    ylab(" ") + theme(axis.text=element_text(size=12),axis.title=element_text(size=16,face="bold"))
```



This perspective takes into account the age of the other figher as well, and as we see, typically the younger the fighter compared to the other, the better chance they hold in beating their opponent, holding all else equal. One might expect experience to play a larger role, but it seems that advantages of youth overrides experience in this data.

Let's try to trim our regression a little bit and put our predictive variables in terms of comparative difference:

```
dat$avg_diff_body_landed <-dat$R_avg_BODY_landed-dat$B_avg_BODY_landed
dat$avg_diff_head_landed <-dat$R_avg_HEAD_landed-dat$B_avg_HEAD_landed
logit2 <- glm(numeric_winner~height_dif+weight_dif+age_diff + avg_diff_body_landed + avg_diff_head_lande
summary(logit2)
```

```
##
## Call:
## glm(formula = numeric_winner ~ height_dif + weight_dif + age_diff +
##     avg_diff_body_landed + avg_diff_head_landed, family = "binomial",
##     data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9854  -1.3179   0.7756   0.9214   1.3795
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.719839   0.036577  19.680   <2e-16 ***
## height_dif           -0.001763   0.005801  -0.304   0.7613
## weight_dif            0.001713   0.002952   0.580   0.5617
## age_diff             -0.074710   0.007289 -10.250   <2e-16 ***
## avg_diff_body_landed  0.004666   0.006171   0.756   0.4496
```

```
## avg_diff_head_landed  0.006702   0.002598   2.579   0.0099 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4592.8  on 3591  degrees of freedom
## Residual deviance: 4456.5  on 3586  degrees of freedom
## AIC: 4468.5
##
## Number of Fisher Scoring iterations: 4
```

So as we see from the significance codes, the things that we are the most confident affect the outcome of a fight is the difference in age and the difference in headshots landed.

Predicting Winner Using Decision Tree

we decided to test our Logistic regression using a Decision Tree to predict the winner

Step 1:Split data in train and test data

We decided to split the data using 0.7 ratio and divide it into two data sets which are training and testing set.

```r
#install.packages("caTools")
library(caTools)
#install.packages("rpart")
library(rpart)
set.seed(2447)
#splitting the data
split <- sample.split(dat, SplitRatio = 0.7)
split
```

```
##   [1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE
##  [25] FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [49]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE
##  [61]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [73]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
##  [85] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
##  [97]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [109]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
## [121]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [133]  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
## [145]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [157]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
```

```r
#dividing the data into trainig and testing subsets
train <- subset(dat, split=="TRUE")
test <- subset(dat, split=="FALSE")
#str(train)
#str(test)
```

Step 2:Train model with logistics regression using glm function

In this we built our model inform of logistic regression and used the train subset as our data.

```r
dmodel <- rpart(numeric_winner~., data=train, method="class")
dmodel
```

```
## n= 2503
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 2503 849 1 (0.3391930 0.6608070)
##   2) Winner=Blue 849   0 0 (1.0000000 0.0000000) *
##   3) Winner=Red 1654   0 1 (0.0000000 1.0000000) *
```

```r
summary(dmodel)
```

```
## Call:
## rpart(formula = numeric_winner ~ ., data = train, method = "class")
##   n= 2503
##
##     CP nsplit rel error xerror       xstd
## 1 1.00      0         1      1 0.02789867
## 2 0.01      1         0      0 0.00000000
##
## Variable importance
##                Winner R_total_rounds_fought              R_losses
##                    93                     2                     2
##                 R_age                R_wins              age_diff
##                     2                     1                     1
##
## Node number 1: 2503 observations,    complexity param=1
##   predicted class=1  expected loss=0.339193  P(node) =1
##     class counts:   849  1654
##    probabilities: 0.339 0.661
##   left son=2 (849 obs) right son=3 (1654 obs)
##   Primary splits:
##       Winner                   splits as  LR,           improve=1122.05000, (0 missing)
##       Blue_result              splits as  RL,           improve=1122.05000, (0 missing)
##       Red_result               splits as  LR,           improve=1122.05000, (0 missing)
##       R_avg_opp_SIG_STR_landed < 17.76786 to the right, improve=  43.56475, (0 missing)
##       B_avg_SIG_STR_att        < 47.10556 to the right, improve=  39.08395, (0 missing)
##   Surrogate splits:
##       R_total_rounds_fought < 40.5     to the right, agree=0.668, adj=0.022, (0 split)
##       R_losses              < 7.5      to the right, agree=0.667, adj=0.019, (0 split)
##       R_age                 < 37.5     to the right, agree=0.666, adj=0.016, (0 split)
##       R_wins                < 13.5     to the right, agree=0.665, adj=0.013, (0 split)
##       age_diff              < 11.5     to the right, agree=0.664, adj=0.008, (0 split)
##
## Node number 2: 849 observations
##   predicted class=0  expected loss=0  P(node) =0.339193
##     class counts:   849     0
##    probabilities: 1.000 0.000
##
```

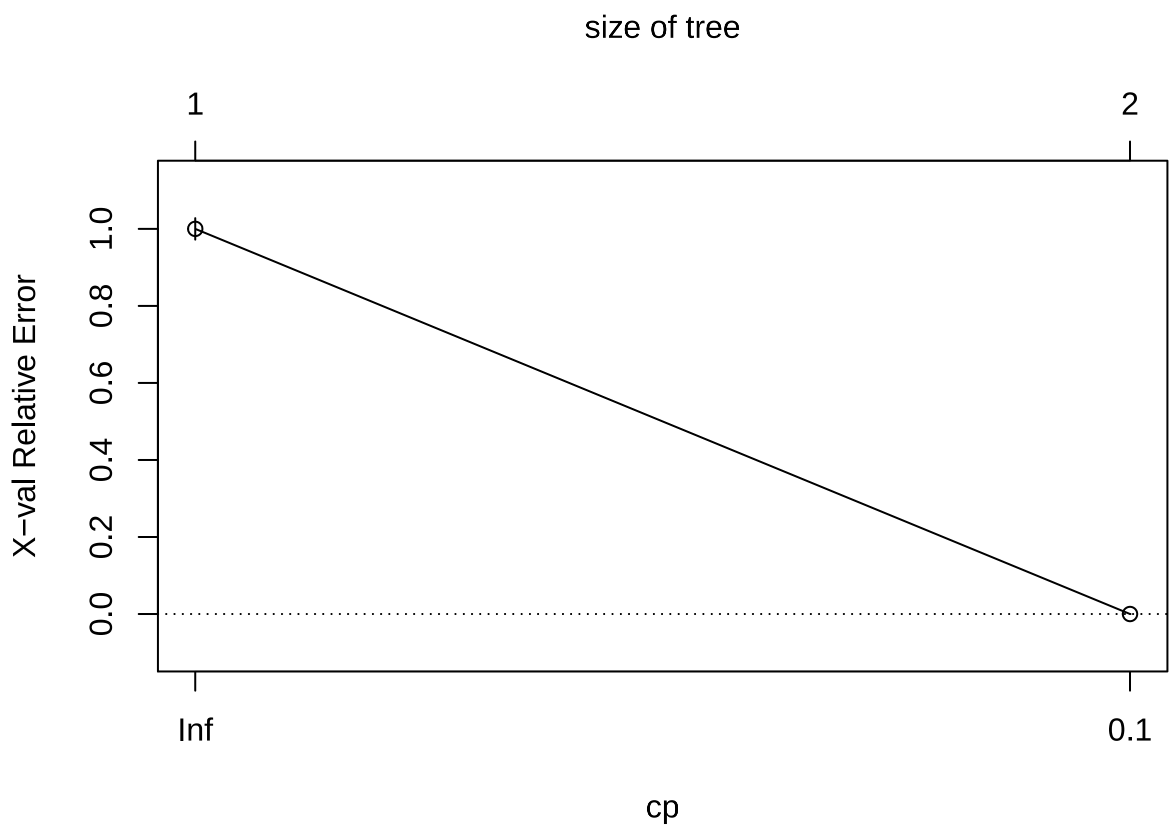
```
## Node number 3: 1654 observations
##   predicted class=1  expected loss=0  P(node) =0.660807
##     class counts:     0  1654
##    probabilities: 0.000 1.000
```

#To visualize how our model is predicting we decided to plot our model From our results we can see the we have split errors hence our decison tree need pruning

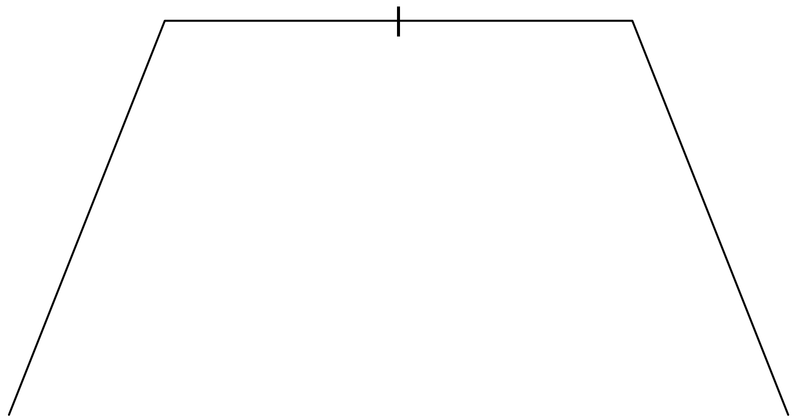
```r
printcp(dmodel)
```

```
##
## Classification tree:
## rpart(formula = numeric_winner ~ ., data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] Winner
##
## Root node error: 849/2503 = 0.33919
##
## n= 2503
##
##     CP nsplit rel error xerror     xstd
## 1 1.00      0         1      1 0.027899
## 2 0.01      1         0      0 0.000000
```

```r
plotcp(dmodel)
```



```r
plot(dmodel,uniform=TRUE,branch=0.6,margin=0.1)
```



Step 3:Predict test data based on trained model we used our trained model to predict the winner on the test data.

```r
test$numeric_winner_predicted <-predict(dmodel, newdata=test, type="class")
table(test$numeric_winner,test$numeric_winner_predicted)
```

```
##
##       0   1
##   0 363   0
##   1   0 726
```

```r
install.packages("caret")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
install.packages("e1071")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```r
library(e1071)
```

Step 4: Evauate Model Accuracy using Confusion matrix

we used confusion matrix to evalute the accuracy of our model

```r
confusionMatrix(table(test$numeric_winner,test$numeric_winner_predicted))
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0 363   0
##   1   0 726
##
##                Accuracy : 1
##                  95% CI : (0.9966, 1)
```

```
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3333
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

Tree Pruning

#Find the value of CP for which cross validation error is minimum

```r
min(dmodel$cptable[,"xerror"])
```

```
## [1] 0
```

```r
which.min(dmodel$cptable[,"xerror"])
```

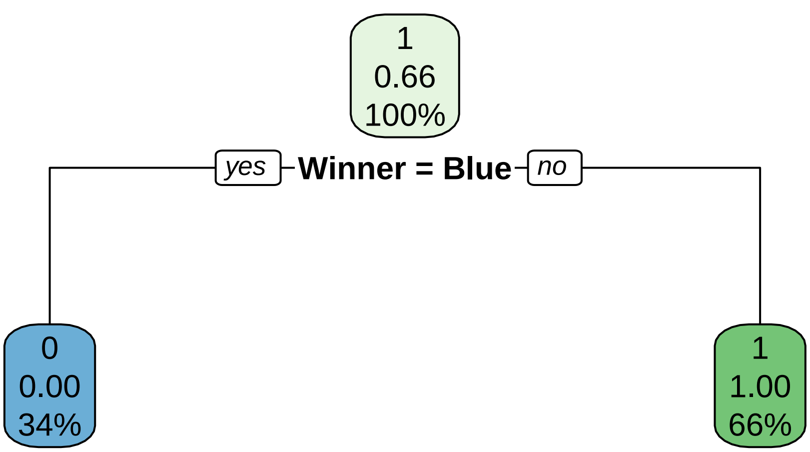
```
## 2
## 2
```

```r
cpmin <- dmodel$cptable[2, "CP"]
```

```r
##install.packages('rpart.plot')
library(rpart.plot)
```

#Prune the tree by setting the CP parameter as = cpmin

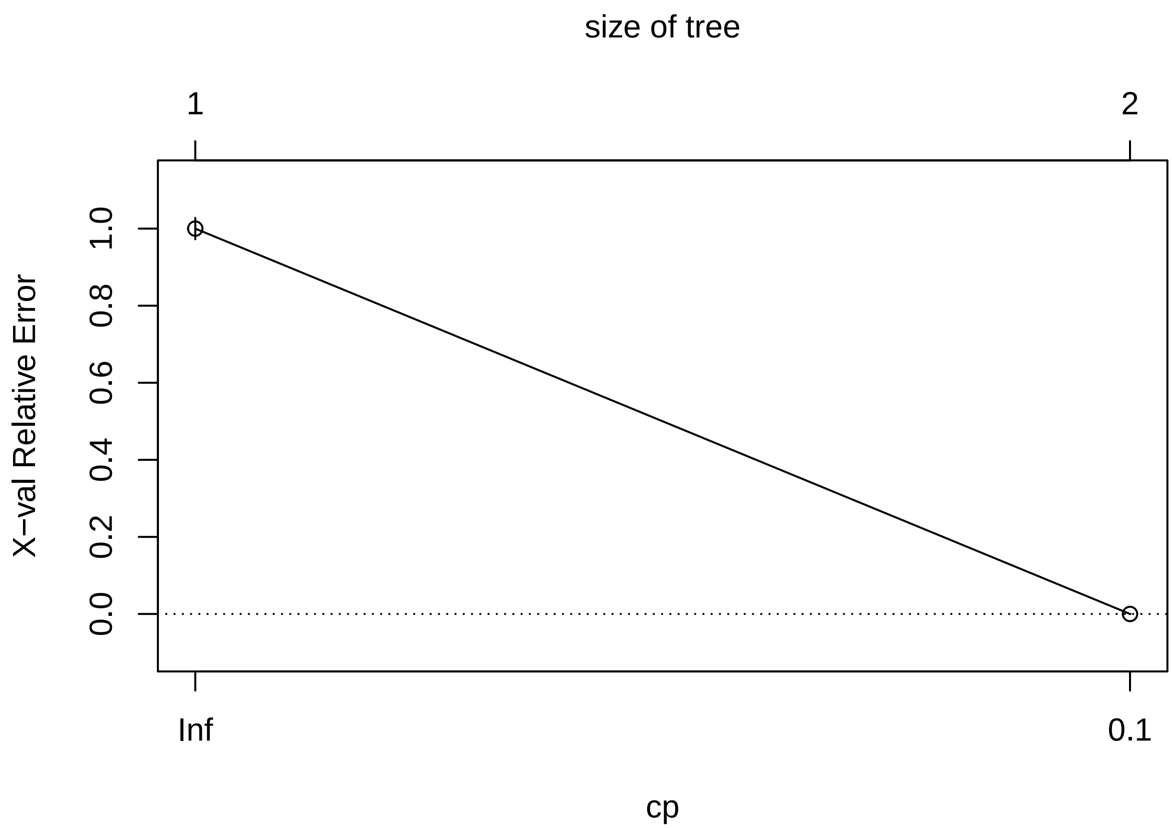
```r
decision_tree_pruned = prune(dmodel, cp = cpmin)
rpart.plot(decision_tree_pruned)
```



```r
printcp(decision_tree_pruned)
```

```
##
## Classification tree:
## rpart(formula = numeric_winner ~ ., data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] Winner
##
## Root node error: 849/2503 = 0.33919
##
## n= 2503
##
##     CP nsplit rel error xerror     xstd
## 1 1.00      0         1      1 0.027899
## 2 0.01      1         0      0 0.000000
```

```r
plotcp(decision_tree_pruned)
```



Predict test data based on trained model

```r
test$numeric_winner_predicted <-predict(decision_tree_pruned, newdata=test, type="class")
table(test$numeric_winner,test$numeric_winner_predicted)
```

```
##
##       0   1
##   0 363   0
##   1   0 726
```

Evaluate Model Accuracy using Confusion matrix

```r
confusionMatrix(table(test$numeric_winner,test$numeric_winner_predicted))
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0 363   0
##   1   0 726
##
##                Accuracy : 1
##                  95% CI : (0.9966, 1)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3333
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```