Exercise 3 - Basic Models

WLS, GLM and Logistic

O'Malley

`r Sys.Date()

Table of Contents


```
knitr::opts_chunk$set(echo=T, warning=F, message=F)
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2

## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(tree)

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

Submission

Download the **Ex3_BasicModels_YourLastName.Rmd** R Markdown file and save it with your own last name. Complete all your work in that template file, **Knit** the corresponding Word or PDF file. Your knitted document **must display your R commands**. Submit your knitted homework document. No need to submit the .Rmd file, just your knitted file.

Also, please prepare your R Markdown file with a **professional appearance**, as you would for top management or an important client.

Please, write all your interpretation narratives outside of the R code chunks, with the appropriate formatting and businesslike appearance. I write all my comments inside of the R code chunk to suppress their display until I print the solution, but you should not do this. I will read your submission as a report to a client or senior management. Anything unacceptable to that audience is unacceptable to me.

Setup

This analysis will be done with the **Hitters{ISLR}** basseball player dataset, using AtBat, Hits, Walks, PutOuts, Assists and HmRun as predictors and player **Salary** as the outcome variable. Let's start with an OLS model and we will then test for heteroskedasticity.

```
# Done for you

library(ISLR) # Contains the Hitters dataset

# Enter the commands below in the R Console window, but NOT in the R Markdown
file. Inspect the data and the description of each predictor, to familiarize
yourself with the data

# ?Salaries
# View(Salaries)

# This dataset has several records with omitted data, let's remove them
Hitters=na.omit(Hitters)

# Fit an OLS model to start with
fit.ols <- lm(Salary ~ AtBat+Hits+Walks+PutOuts+Assists+HmRun, data=Hitters)
summary(fit.ols)

##
## Call:
## lm(formula = Salary ~ AtBat + Hits + Walks + PutOuts + Assists +
##     HmRun, data = Hitters)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -920.3 -215.7  -47.7  175.4 2007.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 124.48415   72.75876   1.711 0.088308 .
## AtBat        -2.43104    0.66358  -3.664 0.000302 ***
## Hits          8.98051    1.97223   4.553 8.17e-06 ***
## Walks         6.34231    1.41170   4.493 1.07e-05 ***
## PutOuts       0.25462    0.08960   2.842 0.004847 **
## Assists       0.06698    0.19649   0.341 0.733485
## HmRun         7.02439    3.61990   1.940 0.053418 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 378.8 on 256 degrees of freedom
## Multiple R-squared:  0.311,  Adjusted R-squared:  0.2949
## F-statistic: 19.26 on 6 and 256 DF,  p-value: < 2.2e-16
```

```
# As the output shows, there are 4 significant predictors: AtBat, Hits, Walks
and PutOuts, and 2 non-significant predictors: Assists and HmRun.
```

1. Heteroskedasticity Testing

1.1 Conduct a **Breusch-Pagan** test for Heteroskedasticity for the **fit.ols** model above.

```
bptest(fit.ols) # Breusch-Pagan test
```

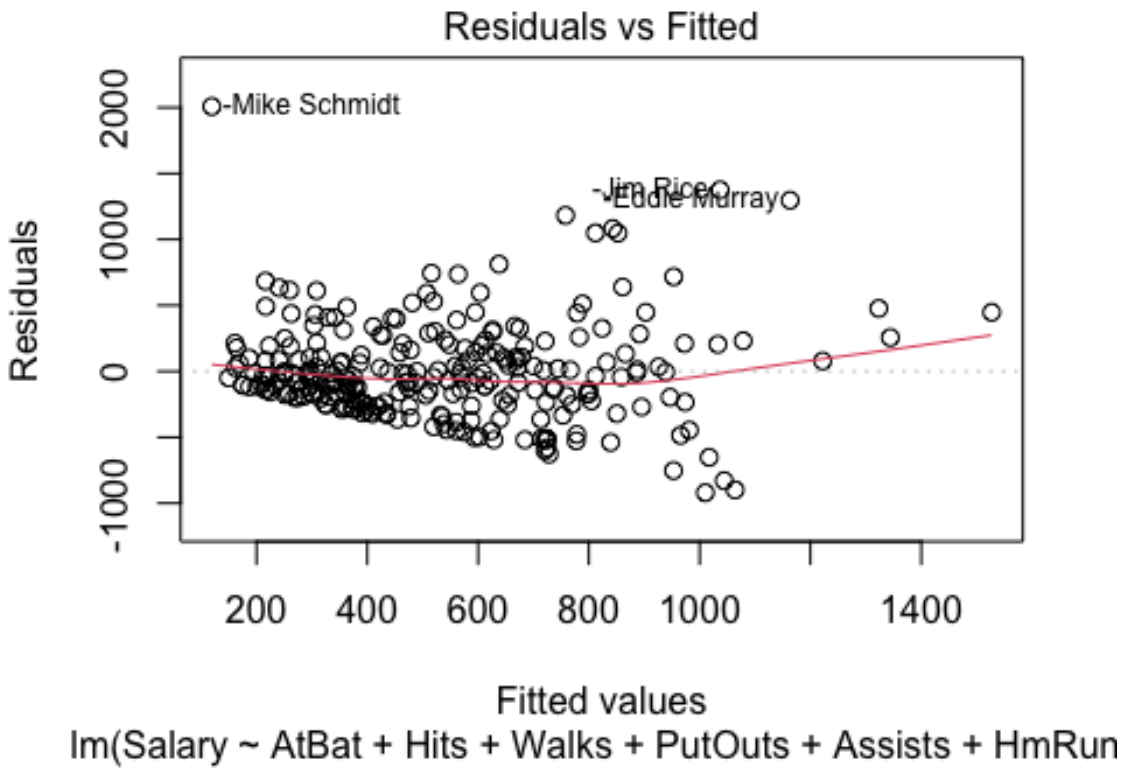
```
##
##  studentized Breusch-Pagan test
##
## data:  fit.ols
## BP = 15.456, df = 6, p-value = 0.01699
```

Analysis

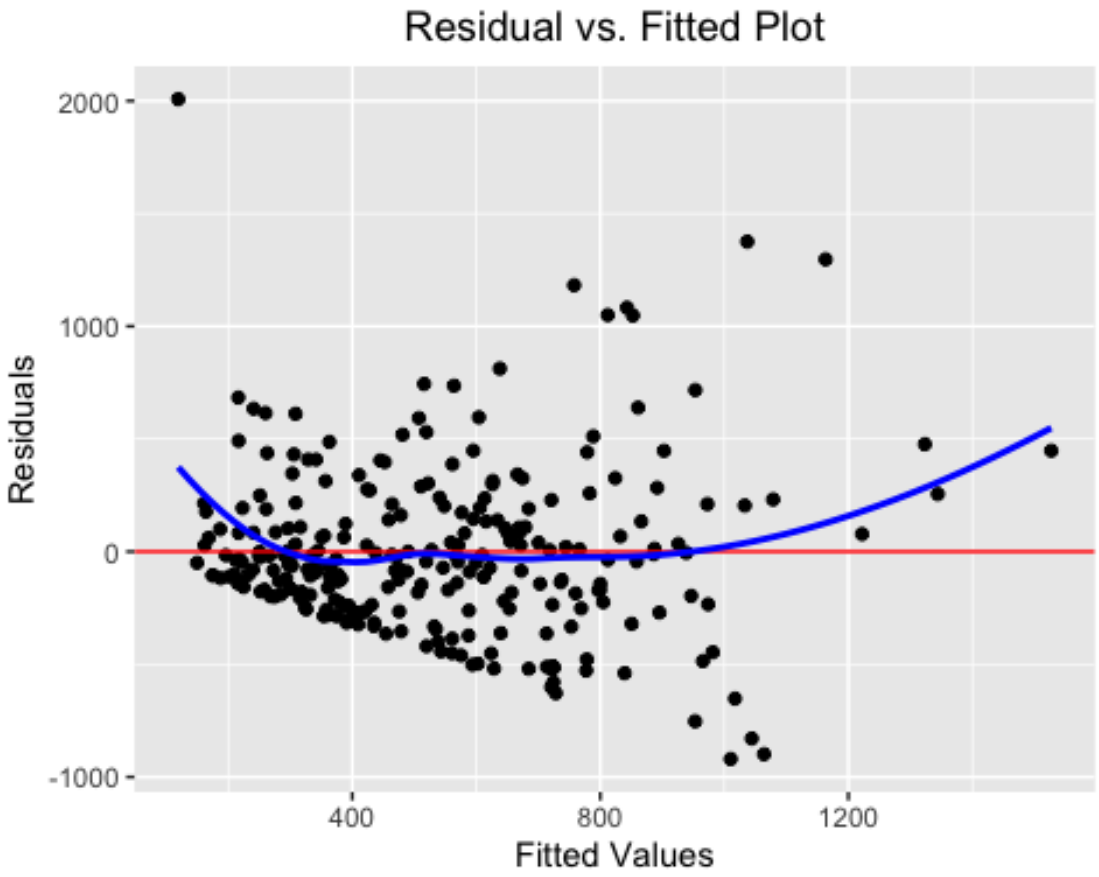
After conducting the Breusch-Pagan test, we can conclude that there is heteroscedasticity in our model. Since p (0.01699) < 0.05, we reject the null hypothesis that homoscedasticity exists in our model.

1.2 Display the first residual plot for **fit.ols** by using `which=1`.

```
plot(fit.ols, which = 1)
```




```r
residuals_df <- data.frame(fitted = fitted(fit.ols), residuals =
resid(fit.ols)) # create data frame
ggplot(residuals_df, aes(fitted, residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  geom_smooth(method = "loess", color = "blue", se = FALSE) +
  labs(x = "Fitted Values", y = "Residuals", title = "Residual vs. Fitted
Plot") +
  theme(plot.title = element_text(hjust = 0.5))
```



1.3 Is there a problem with Heteroskedasticity? Why or why not? In your answer, please refer to **both**, the BP test and the residual plot.

Analysis

Yes, there appears to be heteroscedasticity in the model. The Breusch-Pagan test causes us to reject the $H0$ that homoscedasticity exists in the model because p (0.01699) < 0.05 and the residual plot shows a cone-patterned dispersion of residuals, which supports the existence of heteroscedasticity in the model.

2. Weighted Least Squares (WLS) Model

2.1 Set up the parameters of the WLS model. Use the `abs()` and `residuals()` functions compute the absolute value of the residuals from the OLS model **fit.ols** and store the results in a vector object named **abs.res** . Then use the `fitted()` function to extract the fitted (i.e., predicted) values from **fit.ols** and store the resuts in a vector object named **fitted.ols**. The run an `lm()` model using the predicted values in **fitted.ols** as a predictor of the absolute value of the residuals in **abs.res**.

Technical tip: Because you are using one data vector to predict another data vector, you don't need the `data=` parameter.

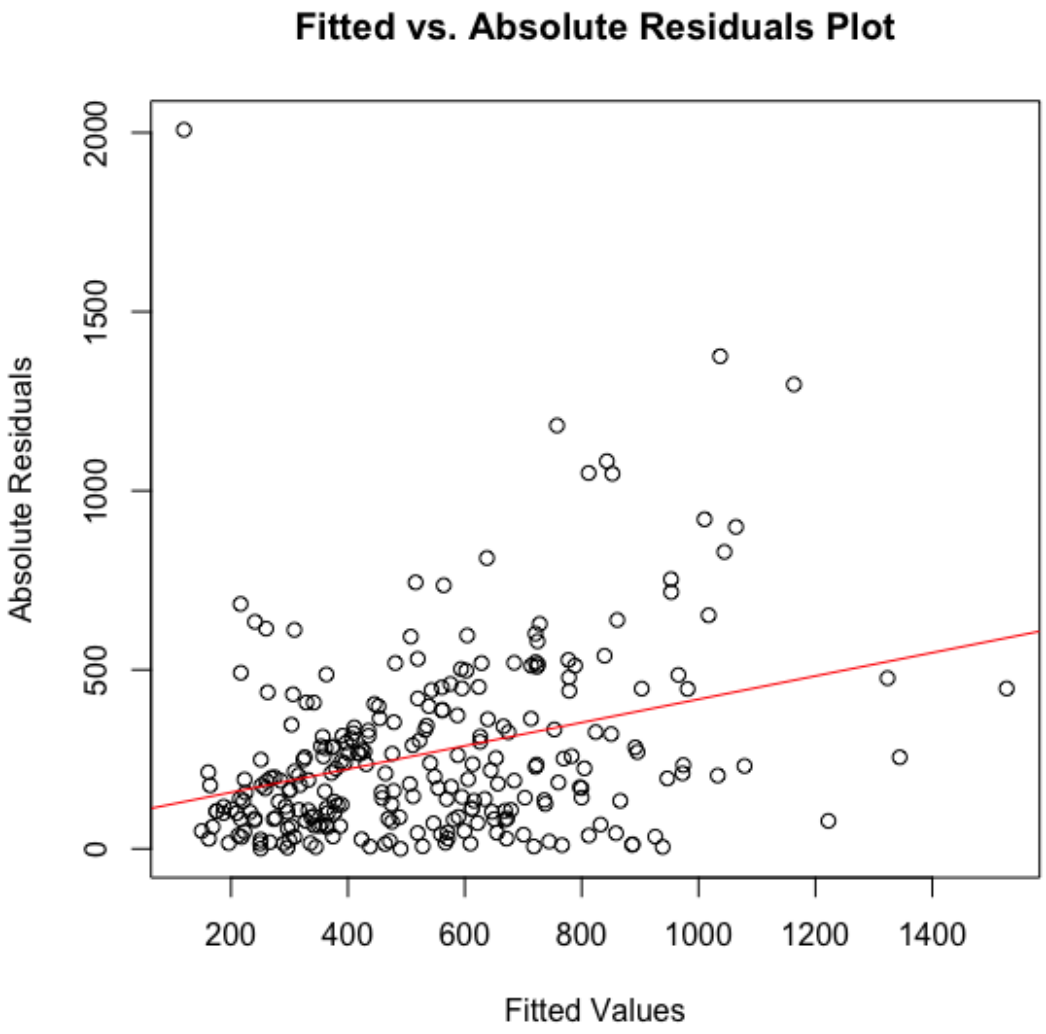
As a sanity check, display the first 10 rows of the `fitted()` values of **lm.abs.res**

```
abs.res <- abs(residuals(fit.ols)) # extract absolute residuals
fitted.ols <- fitted(fit.ols) # extract fitted values

lm.abs.res <- lm(abs.res ~ fitted.ols) # second OLS Model

fitted(lm.abs.res)[1:10] # sanity check
```

```
##       -Alan Ashby      -Alvin Davis     -Andre Dawson -Andres Galarraga
##          270.2217          406.4748          291.4550          285.5487
##  -Alfredo Griffin        -Al Newman  -Argenis Salazar    -Andres Thomas
##          270.8002          149.4738          141.2410          162.5797
##   -Andre Thornton    -Alan Trammell
##          257.5593          342.6095
```

2.2 To visualize the lm.abs.res regression line, plot the **fitted.ols** vector against the **abs.res** vector. Then draw a red line using the `abline()` function for the **lm.abs.res** regression object.

```
plot(fitted.ols, abs.res, xlab = "Fitted Values", ylab =  "Absolute
Residuals", main = "Fitted vs. Absolute Residuals Plot")
abline(lm.abs.res, col = "red")
```



2.3 Specify and Run the WLS Model. First, a vector named **wts** equal to the inverse squared predicted values of **lm.abs.res** (use `wts <- 1/fitted(lm.abs.res)^2`).

Then fit the WLS regression model using the same predictors you used in ols.fit, but using **wts** as the `weights`. Name this regression object **wls.fit**. Display the summary results.

While we are at it, also fit a similar weighted GLM model (**WGLM**), by using the `glm()` function and storing the results in an object named **fit.wglm**. Then display the `summary()` results for the WGLM.

```
wts <- 1/fitted(lm.abs.res)^2 # calculate weights

wls.fit <- lm(Salary ~ AtBat+Hits+Walks+PutOuts+Assists+HmRun, data=Hitters,
weights = wts) # WLS Model
summary(wls.fit) # summary statistics
```

```
##
## Call:
## lm(formula = Salary ~ AtBat + Hits + Walks + PutOuts + Assists +
##     HmRun, data = Hitters, weights = wts)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0512 -1.0607 -0.2793  0.6520 13.6904
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 259.2218    61.2878   4.230 3.26e-05 ***
## AtBat        -2.6758     0.6914  -3.870 0.000138 ***
## Hits          8.4446     2.2715   3.718 0.000247 ***
## Walks         4.4277     1.5889   2.787 0.005723 **
## PutOuts       0.2953     0.1157   2.553 0.011257 *
## Assists       0.4160     0.2022   2.057 0.040679 *
## HmRun        10.4194     4.0230   2.590 0.010150 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.542 on 256 degrees of freedom
## Multiple R-squared:  0.1747, Adjusted R-squared:  0.1554
## F-statistic: 9.034 on 6 and 256 DF,  p-value: 5.81e-09
```

```
fit.wglm <- glm(Salary ~ AtBat+Hits+Walks+PutOuts+Assists+HmRun,
data=Hitters, weights = wts) # WGLM Model
summary(fit.wglm) # summary statistics
```

```
##
## Call:
## glm(formula = Salary ~ AtBat + Hits + Walks + PutOuts + Assists +
##     HmRun, data = Hitters, weights = wts)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 259.2218    61.2878   4.230 3.26e-05 ***
## AtBat        -2.6758     0.6914  -3.870 0.000138 ***
## Hits          8.4446     2.2715   3.718 0.000247 ***
## Walks         4.4277     1.5889   2.787 0.005723 **
## PutOuts       0.2953     0.1157   2.553 0.011257 *
## Assists       0.4160     0.2022   2.057 0.040679 *
## HmRun        10.4194     4.0230   2.590 0.010150 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.378078)
##
##     Null deviance: 737.69  on 262  degrees of freedom
## Residual deviance: 608.79  on 256  degrees of freedom
```

```
## AIC: 3897.9
##
## Number of Fisher Scoring iterations: 2
```

2.3 Observe the similarities an differences between the OLS, WLS and WGLM model and provide a brief commentary of your observations.

Analysis

The r-squared value has decreased from the original OLS model to the WLS, although the WLS model now accounts for heteroscedasticity. Two statistically insignificant variables - Assists and HmRun - have now become significant to both the WLS and WGLM models. Finally, in the WGLM model, Residual Deviance < Null Deviance, which means the model is more accurate than random guessing - a good sign for the WGLM model.

3. Logistic Regression

3.1 Download the **myopia.csv** file to your working directory. Then read it using `read.table()` with the parameters `header=T, row.names=1, sep=","`. Store the dataset in an object named **myopia**.

Dataset documentation at: https://rdrr.io/cran/aplore3/man/myopia.html Please note that **myopic** is coded as 1 (Yes), 0 (No) (not 1 and 2)

For sanity check, list the first 10 rows and 8 columns of this dataset.

```
read.table("myopia.csv", header = TRUE, row.names = 1, sep = ",") -> myopia # assign table to vector

myopia[1:10, 1:8] # sanity check
```

```
##    study.year myopic age female  spheq    al   acd    lt
## 1        1992      1   6      1 -0.052 21.89 3.690 3.498
## 2        1995      0   6      1  0.608 22.38 3.702 3.392
## 3        1991      0   6      1  1.179 22.49 3.462 3.514
## 4        1990      1   6      1  0.525 22.20 3.862 3.612
## 5        1995      0   5      0  0.697 23.29 3.676 3.454
## 6        1995      0   6      0  1.744 22.14 3.224 3.556
## 7        1993      0   6      1  0.683 22.33 3.186 3.654
## 8        1991      0   6      1  1.272 22.39 3.732 3.584
## 9        1991      0   7      0  1.396 22.62 3.464 3.408
## 10       1991      0   6      1  0.972 22.74 3.504 3.696
```

3.2 Fit a logistic model to predict whether a kid is **myopic**, using `age + female + sports.hrs + read.hrs + mommy + dadmy` as predictors. **Tip:** use `family="binomial"(link="logit")`. Store the results in an object named **myopia.logit**. Display the `summary()` results. Then display the `summary()` results.

```r
myopia.logit <- glm(myopic ~ age + female + sports.hrs + read.hrs + mommy +
dadmy, data = myopia, family = binomial(link = "logit")) # logreg model
summary(myopia.logit) # summary statistics
```

```
##
## Call:
## glm(formula = myopic ~ age + female + sports.hrs + read.hrs +
##     mommy + dadmy, family = binomial(link = "logit"), data = myopia)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.83835    2.21222  -3.995 6.46e-05 ***
## age         -0.03073    0.29219  -0.105    0.916
## female      -0.15787    0.46980  -0.336    0.737
## sports.hrs  -0.13993    0.03507  -3.990 6.60e-05 ***
## read.hrs     0.79920    0.09929   8.049 8.35e-16 ***
## mommy        2.93733    0.54288   5.411 6.28e-08 ***
## dadmy        2.77087    0.54069   5.125 2.98e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 480.08  on 617  degrees of freedom
## Residual deviance: 131.00  on 611  degrees of freedom
## AIC: 145
##
## Number of Fisher Scoring iterations: 8
```

3.3 For interpretation purposes, display the log-odds alongside the odds. Use the `coef()` function to extract the log-odds coefficients from **myopia.logit** and save them in a vector object named **log.odds**. Then use the `exp()` function to convert the log-odds into odds and store the results in a vector object named **odds**. The enter the `options(scipen=4)` command to minimize the use of scientific notation. Finally, list the log-odds and odds side by side. To do this, use the `cbind()` function to bind the two vectors into one table and name the columns **"Log-Odds"** and **"Odds"** respectively. Embed the `cbind()` function inside the `print()` function with the parameter `digits=2` to get a more compact display.

```r
log.odds <- coef(myopia.logit) # logodds coefficients
odds <- exp(log.odds) # calculate odds
print(cbind("Log-Odds" = log.odds, "Odds" = odds), digits = 2)
```

```
##             Log-Odds    Odds
## (Intercept)   -8.838 1.5e-04
## age           -0.031 9.7e-01
## female        -0.158 8.5e-01
## sports.hrs    -0.140 8.7e-01
## read.hrs       0.799 2.2e+00
## mommy          2.937 1.9e+01
## dadmy          2.771 1.6e+01
```

3.4 Provide a brief interpretation of both, the log-odds and odds effects of read.hrs and mommy. Please refer to the respective variable measurement units in your discussion.

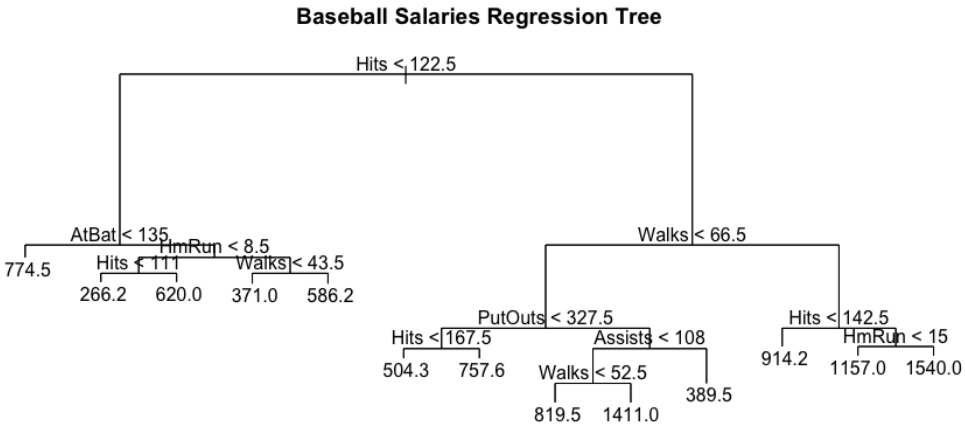
Analysis

On average, holding everything else constant, there is a positive relationship between read.hrs and myopia (Log-Odds = 0.799) and a one unit increase in read.hrs multiplies the odds of myopia by 2.22 (increase of 122.2%). On average, holding everything else constant, there is a stronger positive relationship between mommy and myopia (Log-Odds = 2.937) and since mommy is a binary variable, if the value is 1, then the odds of myopia are multiplied by 18.86 (increase of 1787.5%).

4. Decision Trees

4.1 Regression Tree. Load the **{tree}** library. Then fit the regression model **ols.fit** above, but this time as a **regression tree** using the `tree()` function and save the results in an object named **fit.tree.salary**. Then plot the tree using the `plot()` and `text()` functions (use the `pretty=0` parameter). Also use the `title()` function to title you tree diagram **"Baseball Salaries Regression Tree"**.

```
tree(fit.ols) -> fit.tree.salary # regression tree model
# plot creation
plot(fit.tree.salary)
text(fit.tree.salary, pretty = 0)
title("Baseball Salaries Regression Tree")
```



4.2 Classification Tree. Fit the Logistic model **myopia.logit**, but this time as a **classification tree** using the `tree()` function and save the results in an object named **fit.tree.myopia**. Then plot the tree using the `plot()` and `text()` functions (use the `pretty=0` parameter). Also use the `title()` function to title you tree diagram **"Myopia Classification Tree"**.

```r
fit.tree.myopia <- tree(myopia.logit) # classification tree model

# plot creation
plot(fit.tree.myopia)
text(fit.tree.myopia, pretty = 0)
title("Myopia Classification Tree")
```