Exercise 3 - Basic Models

WLS, Logistic and Trees

Dawoon (Kate) Jung

2021-02-07

Table of Contents

knitr::opts\_chunk$set(echo=T, warning=F, message=F)

## 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}** baseball player data set, 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.

# Prep work done for you  
  
library(ISLR) # Contains the Hitters data set  
  
# 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  
  
?Hitters  
View(Hitters)  
  
# This data set 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

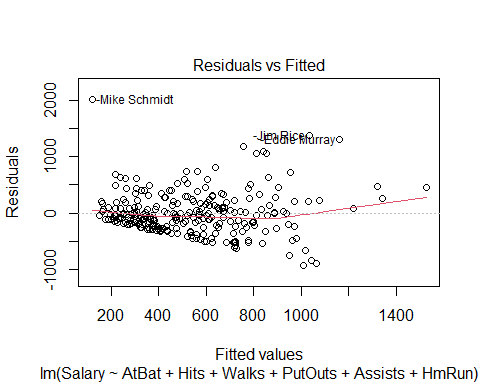
1.1 Load the {lmtest} library and conduct a **Breusch-Pagan** test for Heteroskedasticity for the **fit.ols** model above, using the bptest() function.

library(lmtest)  
  
bptest(fit.ols, data = Hitters)

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

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

plot(fit.ols, which = 1)



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.

# There is heteroskedasticity. The p-value of the Breush-Pagan test is 0.01699, which is referring the significance of heteroskedasticity. According to the first residual plot, the error variance is not even, has some pattern.

## 2. Weighted Least Squares (WLS) Model

2.1 Set up the parameters of the WLS model. Using 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**. Then fit 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.

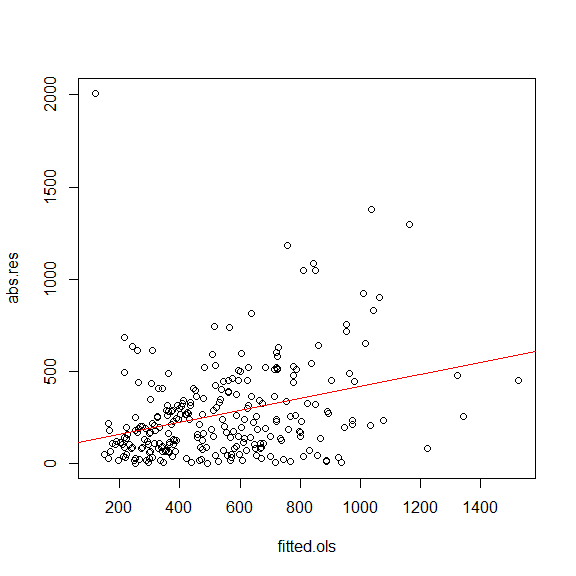
As a sanity check, display the first 10 rows of the fitted() values of **lm.abs.res**

abs.res <- abs(residuals(fit.ols))  
fitted.ols <- fitted(fit.ols)  
lm.abs.res <- lm(abs.res ~ fitted.ols)  
  
fitted(lm.abs.res)[1:10]

## -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)  
abline(lm.abs.res, col = "red")



2.3 Specify and Run the WLS Model. First, create a weight 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 parameter. 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)  
wts[1:10]

## -Alan Ashby -Alvin Davis -Andre Dawson -Andres Galarraga   
## 1.369493e-05 6.052473e-06 1.177218e-05 1.226421e-05   
## -Alfredo Griffin -Al Newman -Argenis Salazar -Andres Thomas   
## 1.363647e-05 4.475790e-05 5.012777e-05 3.783270e-05   
## -Andre Thornton -Alan Trammell   
## 1.507458e-05 8.519245e-06

fit.wls <- lm(Salary ~ AtBat + Hits + Walks + PutOuts + Assists + HmRun, data = Hitters, weights = wts)

fit.wglm <- glm(Salary ~ AtBat + Hits + Walks + PutOuts + Assists + HmRun, data = Hitters, weights = wts)

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

summary(fit.wls)

##   
## 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

summary(fit.wglm)

##   
## Call:  
## glm(formula = Salary ~ AtBat + Hits + Walks + PutOuts + Assists +   
## HmRun, data = Hitters, weights = wts)  
##   
## Deviance 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  
##   
## (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

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

# In many cases, the R squared for WLS is higher than the one for OLS, but in this dataset, the R squared for WLS is smaller. (0.311 > 0.1747) However, comparing these two is not a proper decision since R squared for WLS is for the proportion of the explained weighted variance, not the variance. As we found out above, since this OLS's residuals are heteroskedastic, WLS has less variance than OLS. WGLM model shows the same result as WLS, but different measures with 2LL fir statistics.

## 3. Logistic Regression

3.1 Download the **myopia.csv** file to your working directiory. 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.

myopia <- read.table("myopia.csv", header = T, row.names = 1, sep = ",")  
myopia[1:10, 1:8]

## 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.

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

##   
## Call:  
## glm(formula = myopic ~ age + female + sports.hrs + read.hrs +   
## mommy + dadmy, family = binomial(link = "logit"), data = myopia)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.65801 -0.10383 -0.03543 -0.00998 2.63769   
##   
## 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 conver 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.

log.odds <- coef(myopia.logit)  
odds <- exp(log.odds)  
  
options(scipen = 4)  
  
print(cbind("Log-Odds" = log.odds, "Odds" = odds), digits = 2)

## Log-Odds Odds  
## (Intercept) -8.838 0.00015  
## age -0.031 0.96974  
## female -0.158 0.85396  
## sports.hrs -0.140 0.86942  
## read.hrs 0.799 2.22377  
## mommy 2.937 18.86550  
## dadmy 2.771 15.97258

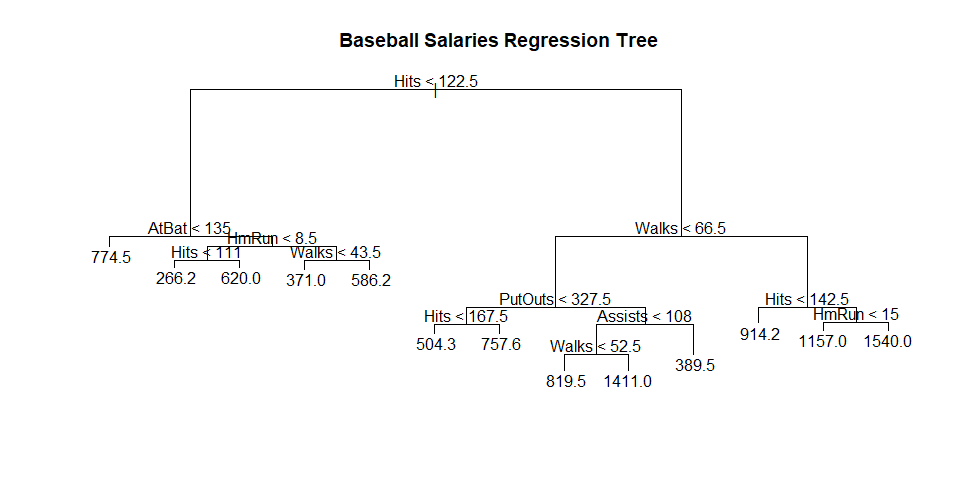
3.4 Provide a brief interpretation of both, the log-odds and odds effects of **read.hrs** and **mommy**. Please refer to the respective variable measurment units in your discussion.

# On average, when the hour of reading per week increases by 1, the log-odds of developing myopia within the first five years goes up by 0.799 when everything else holds constant. In odds, it increases by a factor of 2.22 in the same situation.   
  
# When the child's mother is myopic, on average, the log-odds of the child developing myopic increase by 2.937, and the odds go up by a factor of 18.86, holding everything else constant.

## 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”**.

library(tree)  
fit.tree.salary <- tree(Salary ~ AtBat + Hits + Walks + PutOuts + Assists + HmRun, data=Hitters)  
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”**.

fit.tree.myopia <- tree(myopic ~ age + female + sports.hrs + read.hrs + mommy + dadmy, data=myopia)  
plot(fit.tree.myopia)  
text(fit.tree.myopia, pretty=0)  
title("Myopia Classification Tree")

