ITEC 621 Exercise 3 - Basic Models

WLS, Logistic and Trees

J. Alberto Espinosa

January 28, 2023

Table of Contents


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

General Instructions

Download the **Ex3_BasicModels_YourLastName.Rmd** R Markdown file and save it with your own **last name** and **date**. Complete all your work in that template file.

Knitting: Knit your .Rmd file into a Word, HTML or PDF file. Your knitted document **must display your R commands**. Knitting and formatting is worth up to **3 points** in this and all exercises.

Formatting: Please ensure that all your text narratives are fully visible (if I can't see the text, I can't grade it). Also, please ensure that your **Table of Contents** is visible and properly formatted. 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 in the text area, outside of the R code chunks, with the appropriate formatting and businesslike appearance. **Note:** I write all my interpretation solutions inside of the R code chunk to suppress their display until I print the solution, but don't need to 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.

Important Formatting Tip About the # Tag: Many students submit their knitted file with text narratives embedded in the table of contents and with the text in the main body in large blue font. This is **NOT** proper business formatting. This is the issue: if you want to write comments inside an R code chunk, you need to use the # tag, which tells R that that line should not be executed and it is there as a comment only. However, if you use the # tag in the text area, R Markdown treats this as **Heading 1** text and ## as **Heading 2** text. Heading text will appear in the table of contents and in large blue font in the main text. Please **DO NOT** use # tags in the main text, except for actual headers and sub-headers in your document.

Submission: Submit your knitted homework document in Canvas. There is no need to submit the .Rmd file, just your knitted file.

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. Also, set the `options(scipen = 4)` to minimize the use of scientific notation.

Familiarize yourself with the Hitters data set by entering 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

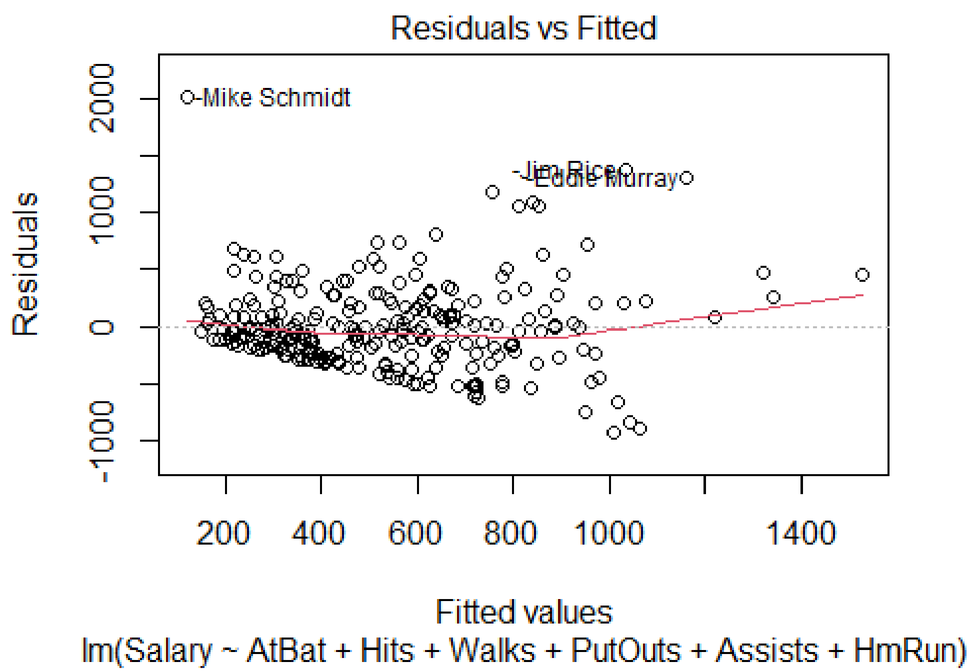
`?Hitters` `View(Hitters)`

Let's start with an OLS model, which you will then test for heteroskedasticity.

```
##
## 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 0.00000817 ***
## Walks         6.34231    1.41170   4.493 0.00001065 ***
## 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
```

1. Heteroskedasticity Testing

1.1 Inspect the residuals visually for heteroskedasticity. To do this, display the first residual `plot()` for **fit.ols** using the parameter `which = 1`.



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

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

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

2. Weighted Least Squares (WLS) Model

2.1 Let's set up the parameters of the WLS model. Let's start by using the `fitted()` function to extract the fitted (i.e., predicted) values from the **fit.ols** object created above and store the results in a vector object named **fitted.ols**.

2.2 Then, 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 `cbind()` function to list the **fitted.ols** and **abs.res** values side by side for the first 10 records (tip: add the index `[1:10, ]` after the function to list only the first 10 rows and all columns)

```
##                   fitted.ols   abs.res
## -Alan Ashby         546.4501  71.45007
## -Alvin Davis        965.4965 485.49645
## -Andre Dawson       611.7531 111.75312
## -Andres Galarraga   593.5884 502.08838
## -Alfredo Griffin    548.2293 201.77066
## -Al Newman          175.0901 105.09010
## -Argenis Salazar    149.7700  49.77005
## -Andres Thomas      215.3972 140.39723
## -Andre Thornton     507.5071 592.49287
## -Alan Trammell      769.0790 251.93600
```

2.3 Now that you have two vectors, one with the absolute value of the residuals and one with the predicted values of the outcome variable Salary, fit an `lm()` model using **fitted.ols** as a predictor vector for the absolute value of the residuals in **abs.res** as the outcome. To check your results, display the first 10 rows of the `fitted()` values of **lm.abs.res** (tip: again, use the `[1:10]` index after the function)

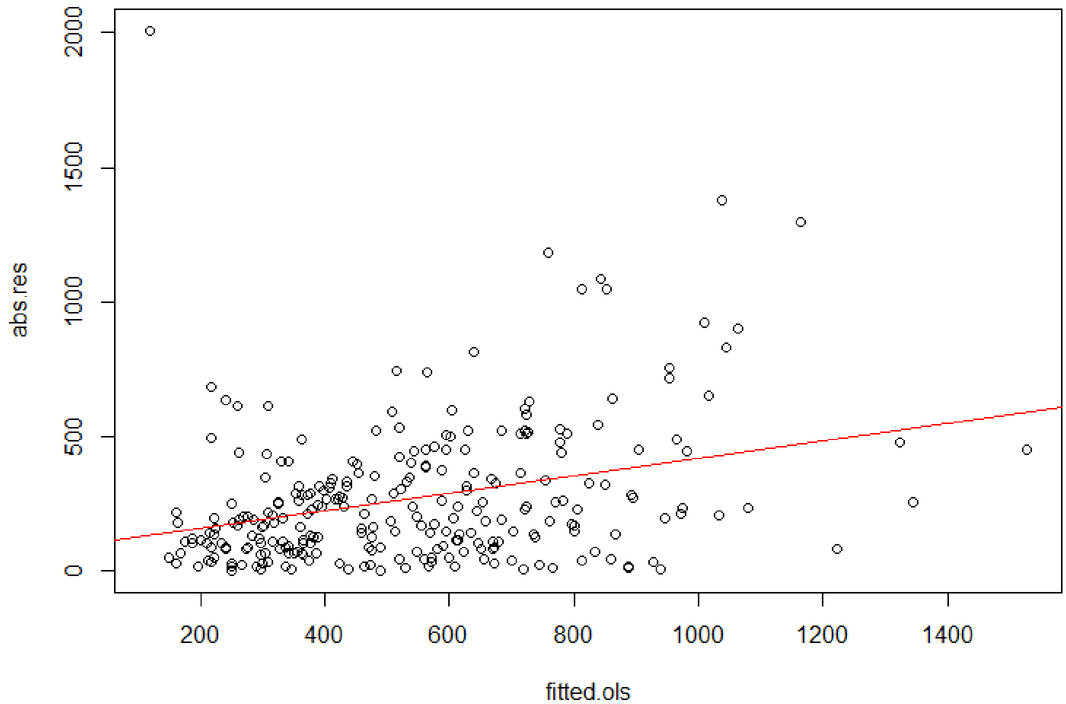
Technical tip: Because you are using one data vector to predict another data vector, you don't need the `data =` parameter. You only need the `data =` parameter when your variables are columns in a data frame.

```
##       -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
```

Think, but no need to answer. What is the difference between **fitted.ols**, **abs.res** and **fitted(lm.abs.res)?

2.4 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.

Technical Note: Notice that I use the `fig.width` and `fig.height` attributes in the `{r` code chunk header to define the size of the plots in inches.



2.5 Specify and run the **WLS** regression model. First, create a weight vector named **wts** equal to the inverse squared predicted values of **lm.abs.res** (tip: use `wts <- 1 / fitted(lm.abs.res) ^ 2`). To check things, display the first 10 rows of the **wts** vector.

```
##       -Alan Ashby       -Alvin Davis      -Andre Dawson -Andres Galarraga
##    0.000013694926     0.000006052473     0.000011772185     0.000012264211
##   -Alfredo Griffin          -Al Newman  -Argenis Salazar     -Andres Thomas
##    0.000013636473     0.000044757897     0.000050127772     0.000037832700
##    -Andre Thornton      -Alan Trammell
##    0.000015074585     0.000008519245
```

Then fit the WLS regression model using the same predictors you used in **ols.fit**, but using **wts** for 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 the exact same specification you used in the `lm()` function, and store the results in an object named **fit.wglm**. Then display the `summary()` results for the WGLM.

```
## 
## 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 0.0000326 ***
## 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
```

```
##
## 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 0.0000326 ***
## 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
```

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

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 data set in an object named **myopia**.

Please review the data set documentation at:
https://rdrr.io/cran/aplore3/man/myopia.html

please note that **myopic** is coded as 1 (Yes), 0 (No), not as 1 and 2.

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

```
##    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 child is **myopic**, using `age + female + sports.hrs + read.hrs + mommy + dadmy` as predictors. Use the parameters `family = "binomial"(link = "logit")` to specify the Logistic model. Store the results in an object named **myopia.logit**. Display the `summary()` results. Then display the `summary()` results.

```
## 
## 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 convert the log-odds into odds and store the results in a vector object named **odds.**

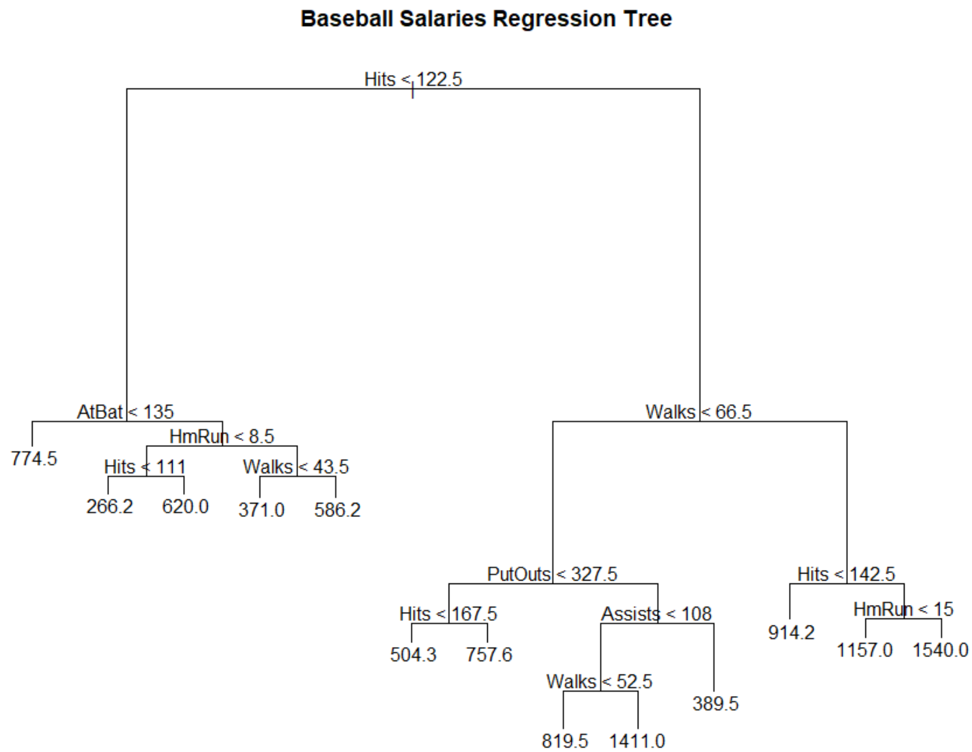
3.4 Finally, list the log-odds and odds side by side using the `cbind()` function. Name the columns as shown in the display below. Once you test that your `cbind()` function is working correctly, embed the function inside the `print()` function with the parameter `digits = 2` to get a more compact display.

```
##             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
```

3.5 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.

4. Decision Trees

4.1 Regression Tree. Load the **{tree}** library. Then fit a regression tree with the same specification as the regression model **ols.fit** above. Use 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**.



4.2 Classification Tree.

Before you start, check the `class()` of the `myopia$myopic` variable and you will notice that it is an integer, not a factor (categorical) variable. This works fine in a Logistic regression model, but a factor outcome variable gives you better visual displays in classification trees. Let's create the corresponding factor variable with `myopia$myopic.f <- as.factor(myopia$myopic)`. Notice that we are renaming the outcome variable so that we don't disturb the original variables. To be certain that the vector was converted from text to factor, list the `class()` of the `myopia$myopic.f` vector.

```
## [1] "integer"

## [1] "factor"
```

Fit a classification tree model using the same specification as the Logistic model **myopia.logit**, but using `myopic.f` as the outcome variable. Use 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**.

Myopia Classification Tree