ITEC 621 Homework 2 - Basic Models and Data Pre-Processing

Kogod School of Business

Prof. J. Alberto Espinosa

February 12, 2023

Table of Contents


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

Knitting, Table of Contents and Presentation (Read Carefully !!)

Download the **HW2_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, HTML 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.

No or Improper knitting, table of contents or inadequate formatting for business can have up to 10 pts. in deductions. So please pay attention to the presentation of your document, including well-written and readable narratives, clean table of contents, visible R code, clear outputs, etc. Overall, your knitted file must have a **professional appearance**, as you would for top management or an important client. Note that while we may deduct up to 10 points for these issues, we may need to deduct more points if we can't understand what you did.

Please, write all your interpretation narratives **outside of the R code chunks** in the areas marked **Answer"**, with the appropriate formatting and businesslike appearance. I write all my comments in the solution inside of the R code chunk with the # tag to suppress (`echo = F`) their display for the homework, which I then turn on (`echo = T`) to knit the solution. I will read your submission as a report to a client or senior management. Anything unacceptable to that audience is unacceptable to me. Write your narratives in the text areas and don't use the # tag, unless your text is a heading.

Specific Instructions

This HW has **5 multi-part questions** related to **basic models** and data **pre-processing**. Each question is worth **20 points**.

1. (20 pts.) Heteroskedasticity and WLS

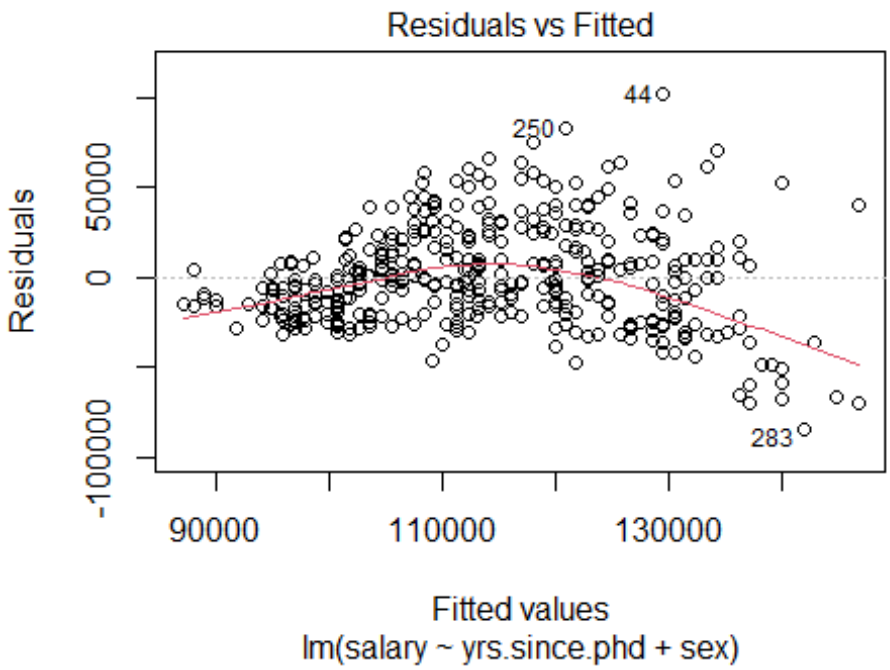
1.1 Load the **{car}** library, which contains the **Salaries** data set (upper case). Then run `options(scipen = 4)` to minimize the use of scientific notation.

Then fit an OLS linear model to predict **salary** (lower case) with **yrs.since.phd** and **sex** as predictors. You will recall that we evaluated this model in Exercise 2. Store the resulting linear model in an object named **fit.ols**. Then display a `summary()` of **fit.ols**.

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -84167 -19735  -2551  15427 102033
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    85181.8     4748.3  17.939   <2e-16 ***
## yrs.since.phd    958.1      108.3   8.845   <2e-16 ***
## sexMale         7923.6     4684.1   1.692   0.0915 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27470 on 394 degrees of freedom
## Multiple R-squared:  0.1817, Adjusted R-squared:  0.1775
## F-statistic: 43.74 on 2 and 394 DF,  p-value: < 2.2e-16
```

As most of you answered correctly in Exercise 2, the coefficient for sexMale has a p-value = 0.0915, which is not significant at the $p < 0.05$ level. It is marginally significant at the $p < 0.10$ providing some evidence of a gender salary gap, but this evidence is weak.

1.2 Now inspect the model for **heteroskedasticity**, first visually and then quantitatively. First, display a residual plot for **fit.ols** using `which = 1`. Then load the **{lmtest}** library and run a **Breusch-Pagan** `bptest()` for heteroskedasticity of the **fit.ols** model above.




```
##
##  studentized Breusch-Pagan test
##
## data:  fit.ols
## BP = 52.797, df = 2, p-value = 3.43e-12
```

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.**

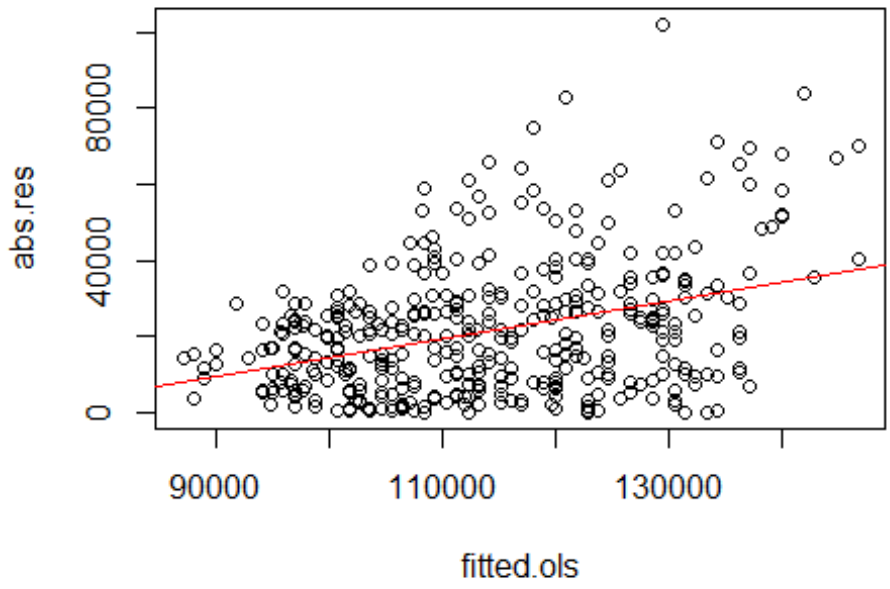
1.4 Given that the residuals of the OLS model are heteroskedastic, fit a Weighted Least Squares **WLS** model. Store this new model in an object named **fit.wls**. Then display the `summary()` results of your WLS model. Let's do this in parts:

(1) First, after you fit **fit.ols** (above), create a vector named **abs.res** with the absolute value of the residual of that model, i.e., `abs(residuals(fit.ols))`;

(2) Then create a vector named **fitted.ols** with the predicted (i.e., fitted) values of the **fit.ols** model, i.e. `fitted(fit.ols)`;

(3) Then, fit a linear model to predict **abs.res** with **fitted.ols** as the predictor. Because these are two objects are vectors already constructed and available in your work environment memory, you don't need the `data =` parameter. Store the results of this model in **lm.abs.res**.

(4) Then `plot()` **fitted.ols** (horizontal axis) against **abs.res** (vertical axis) and layer the **lm.abs.res** regression line on top, colored in red.



(5) Then, take the **square** value of the predicted (i.e., `fitted()`) values of this **lm.abs.res** model and take the inverse (i.e., 1 divided by) of the result and store it the weight vector **wts**. Display the first 10 values of the **wts** vector to double-check your results.

```
##             1             2             3             4             5
6
## 2.477483e-09 2.366100e-09 5.845820e-09 9.618315e-10 1.118038e-09
5.091880e-09
##             7             8             9            10
## 1.570648e-09 9.618315e-10 2.262064e-09 4.027476e-09
```

Then fit a **WLS** model (with the same specification as the OLS model above), using the **wts** weight vector as the `weight =` parameter, and save the model in an object named **fit.wls**. Then display the `summary()` of **fit.wls**

```
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8175 -0.9484 -0.0595  0.8069  3.2642
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      75350       2772  27.181  < 2e-16 ***
## yrs.since.phd     1436        100  14.360  < 2e-16 ***
## sexMale           7952       2858   2.783  0.00565 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.203 on 394 degrees of freedom
## Multiple R-squared:  0.3717, Adjusted R-squared:  0.3686
## F-statistic: 116.6 on 2 and 394 DF,  p-value: < 2.2e-16
```

1.5 Interpretation: Please provide a brief interpretation of what changed from OLS to WLS. More specifically: (1) did the R-squared change? (2) Which model is better, OLS or WLS and why? And (3) Does the WLS model support the gender pay gap hypothesis and why?

2. (20 pts.) Logistic Regression

Dataset: **IBMAttrition.csv** is a Kaggle fictional data set created by IBM

- Attrition (Yes or No): whether the employee left IBM or not
- JobLevel: 1 to 5 (Integer)
- Age (in years)
- BusinessTravel (Factor): "Non-Travel", "Travel_Frequently" or "Travel_Rarely"
- DistanceFromHome (Discrete): Communing miles from home
- JobSatisfaction (Ordinal): 1 (Low); 2 (Medium); 3 (High); 4 (Very-High)
- Gender (Male or Female)
- Marital Status (Factor): "Divorced", "Married", "Single"
- Overtime (Yes or No): whether the employee works overtime or not

2.1 Data Work

In prior examples have used `read.table( )` to read .csv data file. An alternative way to do this is to use `read.csv()`, but you need to be aware of the default parameters of each function. Let's try the `read.csv()` function this time to read the **IBMAttrition.csv** data set and store it in an object named **attr**. As opposed to `read.table()` the defaults on `read.csv()` are `header = T, sep = ","`, so there is no need to enter these parameters. But you need to enter the parameters `row.names =  1, stringsAsFactors = T`. The last parameter is particularly important in R version 4.xx to ensure that the text data is read into factor variables.

After you read the data, get a `summary()` of the data object **attr** to inspect its data types. The summary() output is very large because it summarizes all variables in the data set. So, let's add and index to **attr** to limit the number of variable to display in the summary (i.e., `attr[c("Attrition", "JobLevel", "Age", "Gender", "MaritalStatus", "OverTime")]`.

```
##  Attrition     JobLevel          Age           Gender     MaritalStatus
##  No :1233   Min.   :1.000   Min.   :18.00   Female:588   Divorced:327
##  Yes: 237   1st Qu.:1.000   1st Qu.:30.00   Male  :882   Married :673
##             Median :2.000   Median :36.00                Single  :470
##             Mean   :2.064   Mean   :36.92
##             3rd Qu.:3.000   3rd Qu.:43.00
##             Max.   :5.000   Max.   :60.00
```

```
##  OverTime
##  No :1054
##  Yes: 416
##
##
##
##
```

In the `summary()` above, categorical variables are summarized by categories and quantitative variables by quartiles. Notice in the output that the **JobLevel** variable is an integer. If we model this predictor as is, an integer, its coefficient will represent how much attrition changes when the job level increases by 1 level, which is not very useful or meaningful. We can get more nuanced explanations of level effects if we convert this variable to categorical. Let's create a categorical variable in the **attr** data frame named **attr***JobLevelCat* ∗∗. *We can do this by converting* ∗∗ *attr* **JobLevel** into a **factor** variable using the `as.factor()` and saving the result as a new column in the data frame **attr$JobLevelCat**. List the `class()` of both variables, `attr$JobLevel and attr$JobLevelCat` to verify that the former is an integer and the second one a factor.

```
## [1] "integer"

## [1] "factor"
```

**2.2 Logistic Regression Model

Fit a logistic regression model to predict **Attrition** (upper case A) using **JobLevelCat**, **Age**, **Gender**, **MaritalStatus** and **OverTime** as predictors. Store the glm() regression results in an object named **attr.fit**. Remember to use the attribute `family = binomial(link = "logit")`. Then display the `summary()` results.

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5642  -0.5761  -0.4062  -0.2582   2.8084
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.45483    0.40784  -3.567 0.000361 ***
## JobLevelCat2         -1.10630    0.18999  -5.823 5.78e-09 ***
## JobLevelCat3         -0.50834    0.24026  -2.116 0.034362 *
## JobLevelCat4         -1.64472    0.50612  -3.250 0.001156 **
## JobLevelCat5         -1.05480    0.51726  -2.039 0.041432 *
## Age                  -0.02717    0.01012  -2.683 0.007291 **
## GenderMale            0.23512    0.16094   1.461 0.144054
## MaritalStatusMarried  0.32188    0.22936   1.403 0.160489
## MaritalStatusSingle   1.15926    0.22756   5.094 3.50e-07 ***
## OverTimeYes           1.50573    0.15870   9.488  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1298.6  on 1469  degrees of freedom
## Residual deviance: 1089.2  on 1460  degrees of freedom
## AIC: 1109.2
##
## Number of Fisher Scoring iterations: 5
```

2.3 Model Evaluation

Is this a good model to predict attrition of IBM employees? Use the deviance statistics of the model to answer this question. In particular, comment on whether the predictors included in this model help reduce deviance, relative to the null model.

2.4 Log Odds and Odds

Then use the `coef(), exp() and cbind()` functions to list the coefficients as **log-odds** and **odds** side by side. **Tip:** use `coef()` to read the **attr.fit** coefficients into a vector named **log.odds**. These should be identical to the coefficients listed above. Then use the `exp()` function to convert the **log.odds** vector into an odds vector named **odds**. Then list both of these vectors side by side using the `cbind()` function.

```
##                         log.odds      odds
## (Intercept)          -1.4548272 0.2334407
## JobLevelCat2         -1.1063035 0.3307794
## JobLevelCat3         -0.5083382 0.6014943
## JobLevelCat4         -1.6447170 0.1930672
## JobLevelCat5         -1.0547957 0.3482636
## Age                  -0.0271660 0.9731997
## GenderMale            0.2351152 1.2650545
## MaritalStatusMarried  0.3218843 1.3797251
## MaritalStatusSingle   1.1592571 3.1875644
## OverTimeYes           1.5057305 4.5074453
```

2.5 Interpretation

Provide an interpretation of the **significance** and both, the **log-odds** and **odds** effects of **Age** and **OverTime** on **Attrition**.

3. (20 pts.) Transformations: Categorical Data

3.1 Factor (i.e., Categorical) Variable Levels

In the results above, **MaritalStatus** is a categorical variable. As you know, when you a categorical variable as predictors, R transforms them into one binary variable for each category, but R drops one of them from the model, which becomes the **reference level**. To better understand this, use the `levels()` function to display the levels of **attr$MaritalStatus**, and double check that the category dropped from the model is the first one alphabetically.

```
## [1] "Divorced" "Married"  "Single"
```

3.2 Effect of Marital Status

Based on the levels for **MaritalStatus** (i.e., Divorced, Married or Single), briefly interpret the **significance**, the **log-odds**, and the **odds** effects for **Married** and **Single** employees.

3.3 Re-Valuing (i.e., Re-Shaping)

Re-Shaping JobLevelCat. We will **re-value** the job level categories. This is different than re-leveling. Re-valuing is simply changing the value labels of the categories.

JobLevelCat is a factor variable with 5 levels, from 1 to 5. But when we read a regression output, a variable like JobLevel3 may not mean much to a manager. Let's change these values to something more meaningful (this will not change the results, only the category labels).

Load the **{plyr}** library and use the `revalue()` function to change the values from `"1"` to `"Entry"`, `"2"` to `"Middle"`, `"3"` to `"Senior"`, `"4"` to `"Top"` and `"5"` to `"Executive"`.

Tip: Assign the results to a new variable named `attr$JobLevelPos` (i.e., Job Level Position) using this function:

revalue(attr$JobLevelCat, c("1" = "Entry", "2" = "Middle", "3" = "Senior", "4" = "Top", "5" = "Executive")

Then, use the `levels()` function to display that the factors in this new variable were re-valued properly.

```
## [1] "Entry"     "Middle"    "Senior"    "Top"       "Executive"
```

3.4 Re-Leveling

In the section above we simply changed the values of the JobLevelCat categories. This will cause the first value alphabetically **"Entry"** to be the reference level. Since this is a good reference level, we will leave it as is.

On the other hand, the levels for **MaritalStatus** are not so useful for comparisons. In the model above, **Divorced** is the first **MaritalStatus** category, alphabetically, but it may be more useful to use **Single** as a reference level. Let's `relevel()` this predictor to make **Single** the reference level. Save the re-leveled attribute in a new column in the data frame `attr$MaritalStatusRlv` (Tip: use the parameter `ref = "Single"`). Then display it's `levels()` to ensure that **Single** is the first level.

```
## [1] "Single"   "Divorced" "Married"
```

3.5 Re-Fit the Logistic Model

Now that you have re-shaped and re-leveled the data, re-fit the GLM Logistic model using the new variables **JobLevelPos** and **MaritalStatusRlv** instead of the old ones. Save the results in an object named **attr.fit.rlv**.

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5642  -0.5761  -0.4062  -0.2582   2.8084
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -0.29557    0.35288  -0.838  0.40225
## JobLevelPosMiddle         -1.10630    0.18999  -5.823 5.78e-09 ***
## JobLevelPosSenior         -0.50834    0.24026  -2.116  0.03436 *
## JobLevelPosTop            -1.64472    0.50612  -3.250  0.00116 **
## JobLevelPosExecutive      -1.05480    0.51726  -2.039  0.04143 *
## Age                       -0.02717    0.01012  -2.683  0.00729 **
## GenderMale                 0.23512    0.16094   1.461  0.14405
## MaritalStatusRlvDivorced -1.15926    0.22756  -5.094 3.50e-07 ***
## MaritalStatusRlvMarried  -0.83737    0.17125  -4.890 1.01e-06 ***
## OverTimeYes               1.50573    0.15870   9.488  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1298.6  on 1469  degrees of freedom
## Residual deviance: 1089.2  on 1460  degrees of freedom
## AIC: 1109.2
##
## Number of Fisher Scoring iterations: 5
```

3.6 Interpretation: Job Level Position Effects

Please interpret the effects of the various Job Level Positions, based on this new results. For simplicity, just interpret the significance and **log-odds** effects. No need to interpret the odds effects.

3.7 Interpretation: Marital Status Effect

Inspect the log-odds coefficients (no need to discuss odds effects) in the two models (**attr.fit** and **attr.fit.rlv**) and discuss briefly how the effects of **marital status** have changed in the re-leveled model. Please don't just read off the coefficients, but provide a discussion of what changed and why?

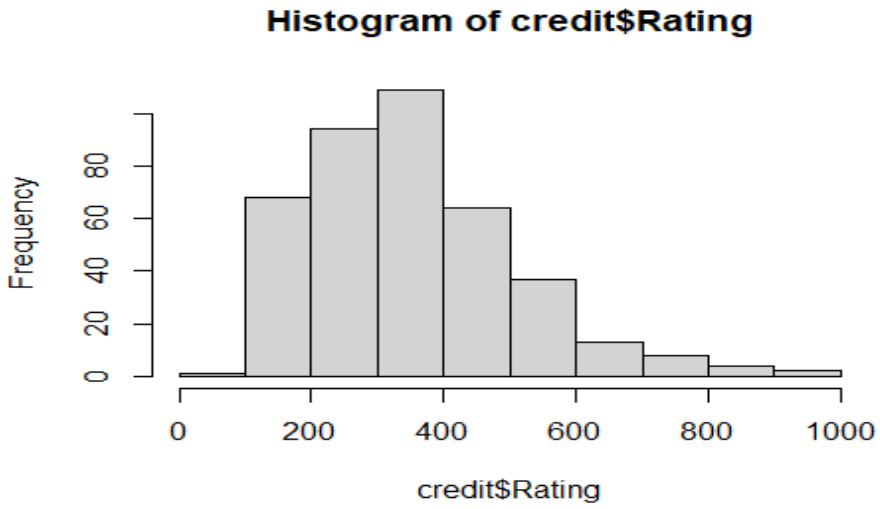
3.8 General Recommendation to IBM

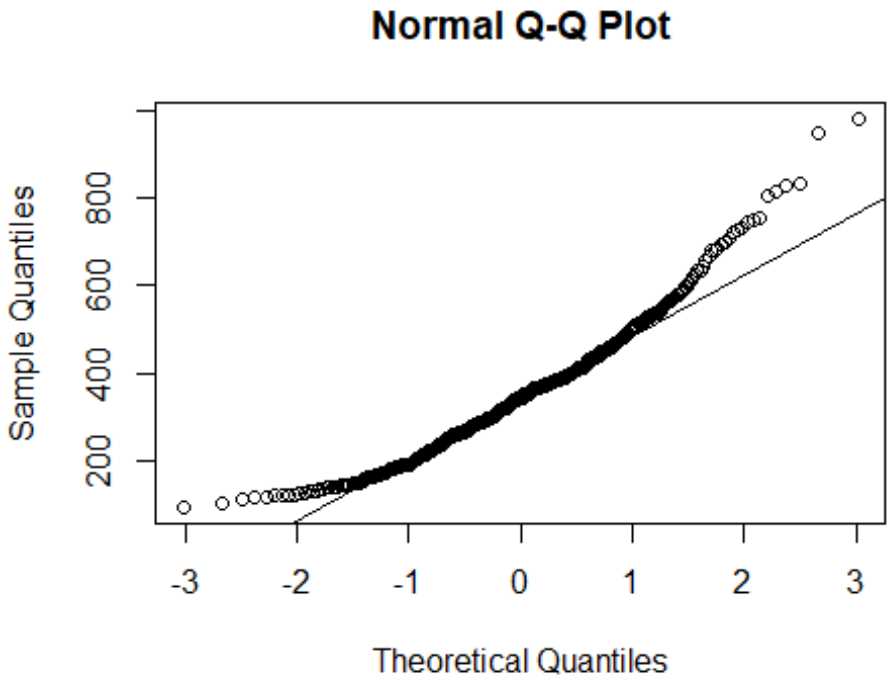
As a business analyst, it is your job to extract meaning from your data and provide an interesting story to your client, supported by your analysis. As is typical, tons of programming scripts, outputs, etc., need to be distilled for management consumption. For this question, simply focus on all the effects observed in the re-leveled model in 3.5 and provide a brief story (6 to 8 lines) that summarizes your interpretations for IBM managers. Provide this interpretation in an English-like narrative for a management audience.

4. (20 pts.) Transformations: Log-Log Model and Standardization

4.1 Using either the `read.table()` or `read.csv` function, read the **Credit.csv** data set into a data frame named **credit**. If you use `read.table()`, ensure that you use `header = T, sep = ",", row.names = 1`. If you use `read.csv()` the only parameter you need is `row.names = 1` . We want to use this data to predict credit **Rating**.

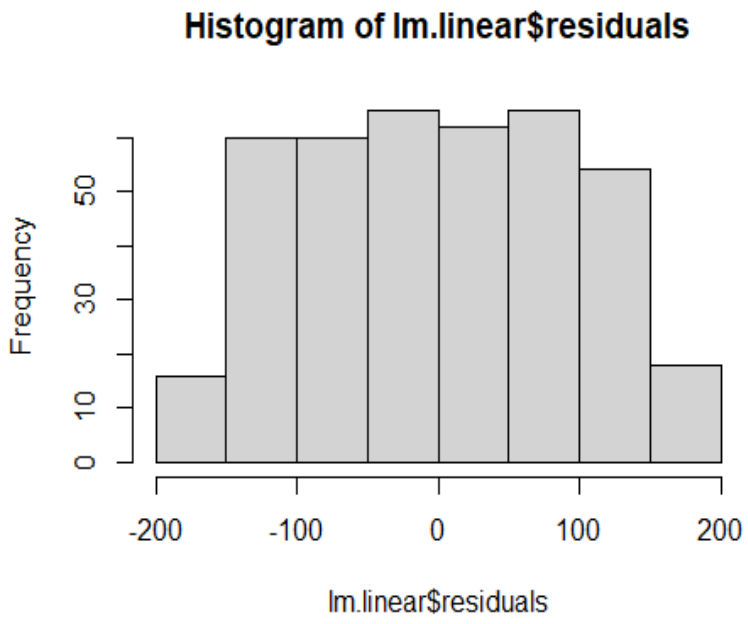
Then display a **histogram** and a **QQ-Plot** for the **Rating** variable. It should be pretty obvious from the histogram that this variable is somewhat skewed to the right.

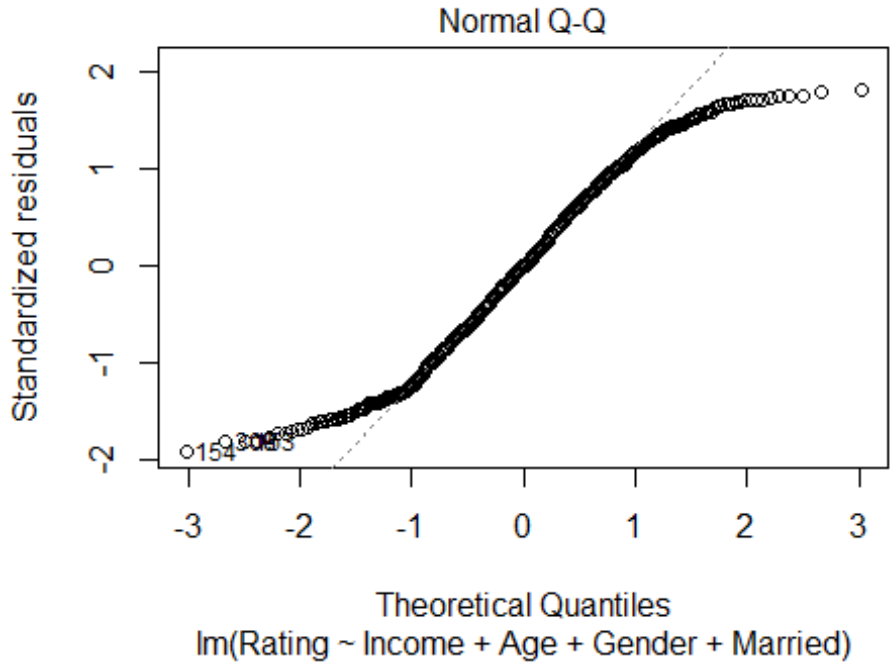




4.2 Given that the response variable is not fully normal, let's start by exploring the normality of the residuals of an OLS model. Fit a **linear** model called **lm.linear** to predict **Rating**, using **Income** (Dollars), **Age**, **Gender** and **Married** as predictors. Display a `summary()` of the results. Then display a histogram of the residuals (tip: stored in `lm.linear$residuals`). Then `plot()` the resulting **lm.linear** model's residual plot, using the `which = 2` parameter.

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -179.032  -77.010   -0.154   79.145  170.117
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  210.8831    18.4526  11.428   <2e-16 ***
## Income         3.5022     0.1370  25.557   <2e-16 ***
## Age           -0.3264     0.2805  -1.163    0.245
## GenderFemale   5.4208     9.4929   0.571    0.568
## MarriedYes     1.7223     9.7741   0.176    0.860
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.86 on 395 degrees of freedom
## Multiple R-squared:  0.6279, Adjusted R-squared:  0.6242
## F-statistic: 166.7 on 4 and 395 DF,  p-value: < 2.2e-16
```





4.3 The residuals look normally distributed in the center of the QQ-Plot and wagging some at the tails. Let's fit a couple of log models to see if we can improve upon the linear model. Please fit both, a **linear-log** model (logging only the predictor variable **Income**; don't log any other variables) and a **log-log** or **elasticity** model , using the same variables as the **linear** model, but logging both the response variable **Rating** and the predictor **Income**

(don't log any other variables). Store the results of the first model in an object named **lm.linear.log** and the second one in an object named **lm.log.log**. Display the `summary()` for both models.

Linear-Log Model

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -206.09  -85.26   -3.06   92.80  363.74
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -221.8455    32.2619  -6.876 2.41e-11 ***
## log(Income)   162.5354     7.8718  20.648  < 2e-16 ***
## Age            -0.1471     0.3164  -0.465    0.642
## GenderFemale    1.7322    10.7231   0.162    0.872
## MarriedYes      8.0376    11.0318   0.729    0.467
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 107.2 on 395 degrees of freedom
## Multiple R-squared:  0.5252, Adjusted R-squared:  0.5204
## F-statistic: 109.2 on 4 and 395 DF,  p-value: < 2.2e-16
```

Log-Log (Elasticity) Model

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90661 -0.20554  0.03478  0.26606  0.68729
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.254779   0.102111  41.668   <2e-16 ***
## log(Income)   0.438578   0.024915  17.603   <2e-16 ***
## Age          -0.001079   0.001001  -1.078    0.282
## GenderFemale  0.013506   0.033939   0.398    0.691
## MarriedYes    0.017270   0.034917   0.495    0.621
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3392 on 395 degrees of freedom
## Multiple R-squared:  0.4432, Adjusted R-squared:  0.4376
## F-statistic: 78.61 on 4 and 395 DF,  p-value: < 2.2e-16
```

4.4 Interpretation: Income is significant in all three models, so no need to discuss significance. But please provide an interpretation of the effect of Income (recorded in thousands of $) on Rating for each of the **three models** fitted above.

4.5 Interpretation: Using the **Adjusted R-Square** as a guide, which of the three models is the best (please note that you **cannot** compare the 3 models with ANOVA because they are **not** nested)

4.6 Then, using the **lm.beta()** function in the **{lm.beta}** library, extract and the standardized regression coefficients for the **lm.linear** model and store them in an object named **lm.linear.std**. Then display its `summary()`.

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -179.032  -77.010   -0.154   79.145  170.117
##
## Coefficients:
##               Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept)  210.88307           NA   18.45262  11.428   <2e-16 ***
## Income         3.50217      0.79775    0.13703  25.557   <2e-16 ***
## Age           -0.32636     -0.03639    0.28054  -1.163    0.245
## GenderFemale   5.42084      0.01753    9.49291   0.571    0.568
## MarriedYes     1.72229      0.00543    9.77408   0.176    0.860
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.86 on 395 degrees of freedom
## Multiple R-squared:  0.6279, Adjusted R-squared:  0.6242
## F-statistic: 166.7 on 4 and 395 DF,  p-value: < 2.2e-16
```

4.8 Interpretation: Briefly interpret the **standardized** effect of **Income** on Rating. Also, briefly answer: is it useful to report or analyze the standardized effect of binary variables like **Gender** or **Married**? Or, is it better to report and discuss the raw unstandardized effect? Why or why not?

5. (20 pts.) Transformations: Lagged Variables and Serial Correlation

For this question, you need to use the **economics** data set contained in the **{ggplot2}** library. Please note that there is a **small issue** in this data set (it has a data frame inside one of the columns), which causes the `slide()` function to give an error. You need to do a simple quick fix to this data set, which is to re-create the data set. I have done this for you below. I also applied the `options()` function to minimize the display of scientific notation. I have done this for you already in the script.

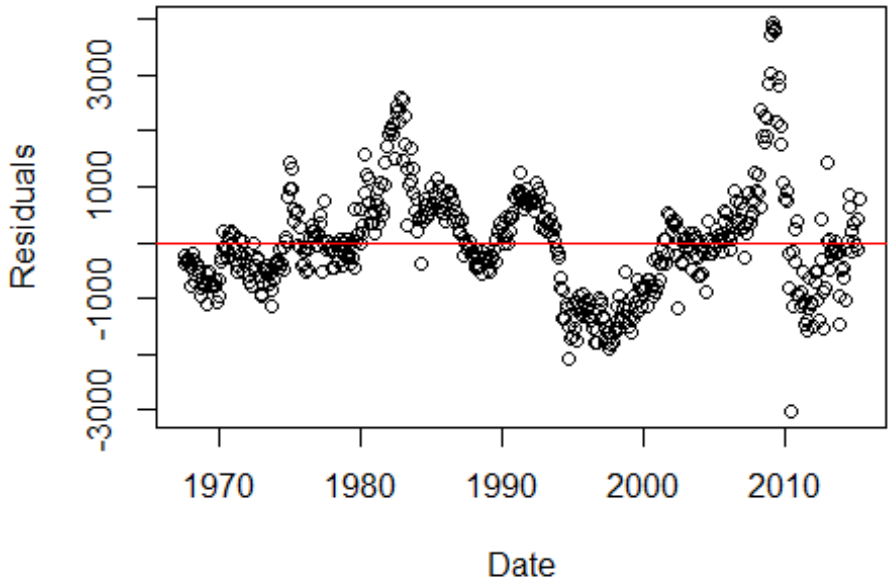
Now, go to the R Console (not in the script) and explore the variables in the `?economics` data set, so that you can interpret results correctly. You will be developing a model to predict **unemployment**.

5.1 Fit a linear model to predict umemployment (**unemploy**, in thousands) as a function of the month of data collection (**date**), personal consumption expenditures (**pce** in billions of dollars), and median duration of unemployment (**uempmed** in weeks). Name this model **fit.linear**. Display the `summary()` result for the resulting linear model.

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3021.3  -546.5   -72.5   499.2  3942.8
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1730.16773  108.99355   15.87   <2e-16 ***
## date           0.85360    0.03895   21.91   <2e-16 ***
## pce           -1.25990    0.05860  -21.50   <2e-16 ***
## uempmed      633.13561   14.50175   43.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 963.3 on 570 degrees of freedom
## Multiple R-squared:  0.8678, Adjusted R-squared:  0.8671
## F-statistic:  1247 on 3 and 570 DF,  p-value: < 2.2e-16
```

5.2 It would appear from the high R-squared that this linear model is good. However, since this is monthly data, it is likely that unemployment in one period may affect unemployment in subsequent periods, so we need to inspect for serial correlation.

Display a scatter plot with `economics$date` (month of the observation) in the horizontal axis and the **residuals** of **fit.linear** (i.e., `fit.linear$residuals`) in the vertical axis. Include the attributes `ylab =''` and `xlab =` to label the vertical and horizontal axes. Also, to help the visual interpretation, also draw a horizontal red line with `abline(0, 0, col="red")`.



Comment: Briefly comment if you suspect serial correlation and why (1 or 2 lines), based on what you see on this plot.

5.3 Now load the **{lmtest}** library and run a Durbin-Wastson test `dwtest()` to confirm or not that the model suffers from serial correlation.

```
##
##  Durbin-Watson test
##
## data:  fit.linear
## DW = 0.18069, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

Comment: Briefly comment if the DW test confirms or not the presence of serial correlation, whether it is positive or negative and why or why not.

5.4 Let's go ahead and correct for serial correlation. My intuition tells me that unemployment in the previous month is a strong predictor of the unemployment this month. Also, I suspect that the unemployment on the same month a year ago may also influence unemployment this month.

So, let's go ahead and load the **{DataCombine}** library and use the `slide()` function to create 2 lagged variables called **unemploy.L1** (lagged 1 month) and **unemploy.L12** (lagged 12 months).

Also, display all columns of the first **15 rows** for the **date** and all three **unemploy** variables and observe how the lag columns were created. Tip, use `economics[1:15, c("date", "unemploy", "unemploy.L1", "unemploy.L12")]`

```
##          date unemploy unemploy.L1 unemploy.L12
## 1  1967-07-01     2944          NA           NA
## 2  1967-08-01     2945        2944           NA
## 3  1967-09-01     2958        2945           NA
## 4  1967-10-01     3143        2958           NA
## 5  1967-11-01     3066        3143           NA
## 6  1967-12-01     3018        3066           NA
## 7  1968-01-01     2878        3018           NA
## 8  1968-02-01     3001        2878           NA
## 9  1968-03-01     2877        3001           NA
## 10 1968-04-01     2709        2877           NA
## 11 1968-05-01     2740        2709           NA
## 12 1968-06-01     2938        2740           NA
## 13 1968-07-01     2883        2938         2944
## 14 1968-08-01     2768        2883         2945
## 15 1968-09-01     2686        2768         2958
```

5.5 Since we don't know whether the unemployment last month, the same month last year or both are influencing the unemployment outcome this year, let's fit 2 lagged models like the linear model above, by adding the predictor **unemploy.L1** in the first model, and both **unemploy.L1** and **unemploy.L12** in the second model. Store the results of the first model

in an object named **fit.lag.1** and the other named **fit.lag.12**. Then test both models for serial correlation with a **Durbin-Watson** test.

```
##
##  Durbin-Watson test
##
## data:  fit.lag.1
## DW = 1.862, p-value = 0.03442
## alternative hypothesis: true autocorrelation is greater than 0

##
##  Durbin-Watson test
##
## data:  fit.lag.12
## DW = 2.0678, p-value = 0.7252
## alternative hypothesis: true autocorrelation is greater than 0
```

Question: Was serial correlation corrected with any of the two lagged models? Why or why not?

5.6 Run a `summary()` of the **fit.lag.12** model and briefly discuss the difference in significance of the predictors and R squared values between the **fit.linear** and **fit.lag.12** models.

Then provide a well-articulated interpretation of the coefficients of the 2 lagged variables in **fit.lag.12**.

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -685.2 -130.1   -3.5  121.4  737.4
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   81.115287  30.180961   2.688  0.00741 **
## unemploy.L1    1.069718   0.009537 112.170  < 2e-16 ***
## unemploy.L12  -0.057289   0.007817  -7.329 8.21e-13 ***
## date          -0.025732   0.012965  -1.985  0.04767 *
## pce            0.048231   0.019027   2.535  0.01152 *
## uempmed      -23.833671   7.662710  -3.110  0.00196 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 198.8 on 556 degrees of freedom
##   (12 observations deleted due to missingness)
## Multiple R-squared:  0.9941, Adjusted R-squared:  0.994
## F-statistic: 1.868e+04 on 5 and 556 DF,  p-value: < 2.2e-16
```