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

Kogod School of Business

Mohamed Seifeldin

Invalid Date

Abstract

This Quarto file contains Homework 2 for ITEC 621, which includes testing for heteroskedasticity, building basic models, like WLS and Logistic regresions, and doing a few data trasformations.

Table of contents

## Rendering (up to 10 pts.)

Open the Quarto template file **HW2\_YourLastName.Qmd**, re-name it using **your actual last name** and copy over your work to the corresponding code chunk sections in the Quarto template.

You are required to **render ALL** your Quarto homework files into a **Word** (preferred) or PDF file. Learning how to build your models in R and report your analysis and results in the same document is an important learning objective of this course. You are expected to submit your homework in a properly rendered document with **business-like** formatting and appearance. **No rendering, inadequate rendering and/or improper formatting of the document will carry point deductions up to 10 points.**

Your **R code** must be **visible** in your rendered document. This means that your Quarto file **MUST** have the attribute echo: true in YAML so that we can evaluate your R code. It is your responsibility to ensure that it is set correctly.

The knitted file must have a table of contents that include all Heading 1 (#) and Heading 2 (##) entries. Please review your Quarto file to ensure that these headings are the **only** text with # or ## tags. The YAML attribute toc: true instructs Quarto to generate a table of contents. The YAML attribute toc-depth: 2 instructs Quarto to include in the table of contents all text prefix with # and ##.

Enter your narrative answers to interpretation questions in the text areas (without # tags), where **Answer:** is noted, not in the R code chunks. It is OK to enter text in the R code chunks with a # tag, but these should be used to make comments and annotations about your script, not for abswers, interpretations or other report narratives.

Also note the YAML attribute code-overflow: wrap above. This attribute makes your code and # marked text to wrap. If you omit this or if you use code-overflow: scroll the text will extend beyond the right margin, which is not what you want in a report.

Overall, anything that would not be acceptable to a business or client audience is not acceptable in rendered documents for this class.

## Interpretations:

Interpretations are the most important part of every homework and assignment in this class. As such, **interpretation questions** will be graded rigorously. Please think through every interpretation question and respond concisely, but accurately. Your analysis must demonstrate that you understand how to interpret the output of your models.

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 will read your submission as a report to a client or senior management. Anything unacceptable to that audience is unacceptable to me.

## 1. (25 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.

Back in Exercise 2, you fitted a simple linear model to test the gender pay gap hypothesis. As we learned in that exercise, the effect of the gender predictor **sex** on **salary** was positive and highly significant, but without other control variables, we suspected that this effect was biased due to omitted variables. I reproduce that simple linear model below.

# Done for you  
library(car)  
options(scipen = 4)  
fit.ols.simple <- lm(salary ~ sex, data = Salaries)  
summary(fit.ols.simple)

Call:  
lm(formula = salary ~ sex, data = Salaries)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-57290 -23502 -6828 19710 116455   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 101002 4809 21.001 < 2e-16 \*\*\*  
sexMale 14088 5065 2.782 0.00567 \*\*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 30030 on 395 degrees of freedom  
Multiple R-squared: 0.01921, Adjusted R-squared: 0.01673   
F-statistic: 7.738 on 1 and 395 DF, p-value: 0.005667

As most of you answered correctly in Exercise 2, the coefficient for sexMale in this simple linear model has a highly significant p-value < 0.001. But we know this effect is biased due to omitted variables.

1.2 To reduce the bias, let’s add some control variables to the model. Fit an OLS linear model to predict **salary** (lower case) with **yrs.service** and **sex** as predictors. Store the resulting linear model in an object named **fit.ols**. Then display a summary() of **fit.ols**.

fit.ols <- lm(salary ~ yrs.service + sex, data = Salaries)  
summary(fit.ols)

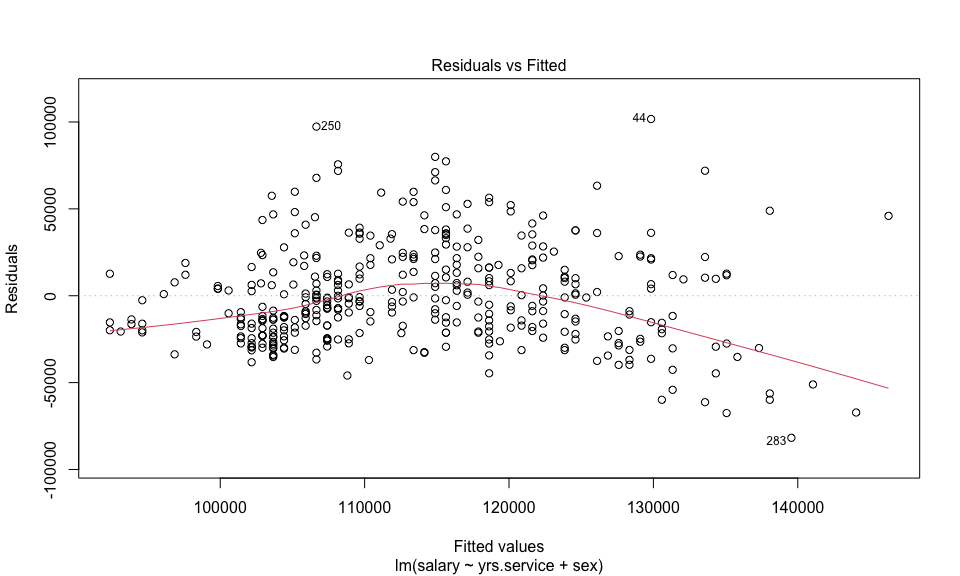
Call:  
lm(formula = salary ~ yrs.service + sex, data = Salaries)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-81757 -20614 -3376 16779 101707   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 92356.9 4740.2 19.484 < 2e-16 \*\*\*  
yrs.service 747.6 111.4 6.711 6.74e-11 \*\*\*  
sexMale 9071.8 4861.6 1.866 0.0628 .   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 28490 on 394 degrees of freedom  
Multiple R-squared: 0.1198, Adjusted R-squared: 0.1154   
F-statistic: 26.82 on 2 and 394 DF, p-value: 1.201e-11

According to this multivariate linear model, once we control for rank and yrs.service, the gender effect becomes only marginally significant at the p = 0.63 level. However, the gender effect is positive and large. We suspect this low statistical confidence in the effect of gender may be caused by heteroskedasticity. That is, while the effect is less biased, it is also also no longer highly significant. If the model suffers from heteroskedasticity we may be able to correct this issue. Let’s investigate

1.2 Let’s inspect the model visually for **heteroskedasticity** and then test it quantitatively.

First, display a residual plot for **fit.ols** using the parameter which = 1. Then load the **{lmtest}** library and run a **Breusch-Pagan** bptest() for heteroskedasticity of the **fit.ols** model above.

library(lmtest)  
plot(fit.ols, which = 1)



bptest(fit.ols)

studentized Breusch-Pagan test  
  
data: fit.ols  
BP = 20.788, df = 2, p-value = 0.00003061

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.** If there is a problem, how does it affect your OLS results (pls. refer specifically to bias and variance in the OLS model and how this may affect the gender gap effect you found)?

**Answer:**

Based on the residual plot, there doesn’t seem to be a clear funnel-shaped pattern, suggesting no obvious presence of heteroskedasticity. Additionally, if the p-value of the Breusch-Pagan test is greater than a chosen significance level (e.g., 0.05), we fail to reject the null hypothesis of homoskedasticity, indicating no evidence of heteroskedasticity.

Heteroskedasticity can lead to biased estimates of the coefficients, particularly the standard errors. This means that the estimated coefficients may not accurately represent the true relationships between the variables. Heteroskedasticity can also inflate the variance of the coefficient estimates, making them less accurate.

1.4 Let’s start developing a WLS model by first running a model to predict the **squared residuals** of fit.ols using the **predicted values** of fit.ols, and using the results to construct the weight vector **wts** to use as a parameter for the related **WLS** model.

**Technical note:** because will be using fitted rather than real values, in a few cases the fitted regression line may briefly cross into the negative site, which will create problems because the weights in WLS cannot be negative. To avoid this problem, all we have to do is force the lm.res2 regression to start at the origin (i.e., have no intercept). This can be easily done by including a **0** as a predictor.

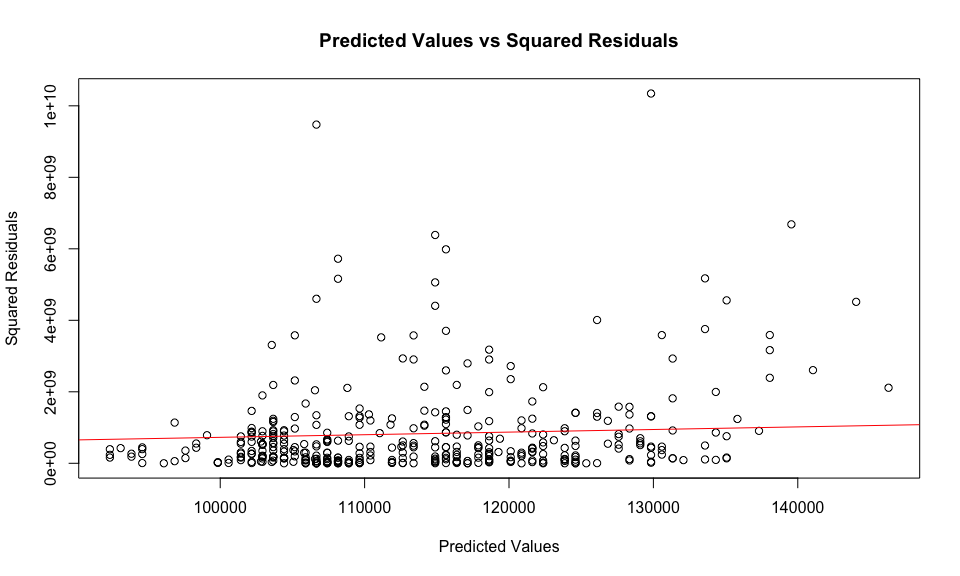
You have a choice of two alternative specifications to do this: one using residuals(fit.ols) ^ 2 as the outcome variables and 0 + fitted(fit.ols) as the predictor; or, alternatively using ted **WLS** model. You have a choice of two alternative specifications to do this: one using fit.ols$residuals ^ 2 as the outcome variables and 0 + fit.ols$fitted.values as the predictor. Your choice. Save the resulting linear model with the name **lm.res2**.

**Technical Note**: normally you would include a data = parameter in the lm() function to indicate the data set to use to fit the model. We don’t need this parameter in the present model because we are regressing a vector on another vector and both vectors are populated with data.

lm.res2 <- lm(residuals(fit.ols) ^ 2 ~ 0 + fitted(fit.ols))  
lm.res2 <- lm(fit.ols$residuals ^ 2 ~ 0 + fit.ols$fitted.values)

Also, to visualize your results, plot() the predictor fitted(fit.ols) against the squared residuals residuals(fit.ols) ^ 2. Alternatively, you could use fit.ols$fitted.values against fit.ols$residuals ^ 2). Your choice. Then draw the corresponding regression line **lm.res2** in red using the function abline(lm.res2, col = "red").

plot(fitted(fit.ols), residuals(fit.ols)^2, xlab = "Predicted Values", ylab = "Squared Residuals",  
 main = "Predicted Values vs Squared Residuals")  
abline(lm.res2, col = "red")



1.5 Now compute the weight vector and fit a Weighted Least Squares **WLS** model with the same outcome and predictors as fit.ols and name it **fit.wls**. First, compute the vector **wts** as the inverse of the predicted values of **lm.res2** 1 / fitted(lm.res2) (or, alternatively 1 / lm.res2$fitted.values) and use this vector object in th weights = parameter of your WLS model. Then display the summary() results of your WLS model.

wts <- 1 / fitted(lm.res2)  
summary(wts)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
9.418e-10 1.140e-09 1.223e-09 1.222e-09 1.310e-09 1.492e-09

Then fit the **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**

fit.wls <- lm(salary ~ yrs.service + sex, data = Salaries, weights = wts)  
summary(fit.wls)

Call:  
lm(formula = salary ~ yrs.service + sex, data = Salaries, weights = wts)  
  
Weighted Residuals:  
 Min 1Q Median 3Q Max   
-2.6452 -0.7222 -0.1183 0.5695 3.5260   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 91054.6 4417.7 20.611 < 2e-16 \*\*\*  
yrs.service 820.4 112.7 7.281 1.81e-12 \*\*\*  
sexMale 9094.7 4547.4 2.000 0.0462 \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.9798 on 394 degrees of freedom  
Multiple R-squared: 0.1377, Adjusted R-squared: 0.1333   
F-statistic: 31.46 on 2 and 394 DF, p-value: 2.104e-13

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

**Answer:**

1. The better model depends on the presence of heteroskedasticity in the data. If heteroskedasticity is present, WLS is preferred over OLS because it corrects for unequal variance by giving more weight to observations with smaller residuals. In this case, since we used the LM test to detect heteroskedasticity and subsequently used WLS to correct for it, WLS is probably the better model as it accounts for the unequal variance.
2. Yes the WLS model does show that their is a stastitacly significant pay gap. the nature of the WLS models is that they use higher weight to the observations which consequently means that the estimates of the coeffiecients are more accurate.

## 2. (25 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 we 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 and higher 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")].

attr <- read.csv("IBMAttrition.csv", row.names = 1, stringsAsFactors = TRUE)  
summary(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 and dummy variables are summarized by categories and quantitative variables by quartiles.

**2.2 Logistic Regression Model**

Fit a logistic regression model to predict **Attrition** (upper case A) using **Age**, **Gender** and a computed variable named **Married** as predictors. The variable **MaritalStatus** has three categorical values: “Single”, “Divorced” and “Married”. For this exercise we are only interested contrasting married against non-married. So, create a new variable named \*\*attr$Married\*\* (note: the `attr$prefix will cause the Married vector to be attached to theattrdata frame). Store in it "Yes" if married and "No" if not (tip: use the functionifelse(attr$MaritalStatus == “Married”, “Yes”, “No”)`).

attr$Married <- ifelse(attr$MaritalStatus == "Married", "Yes", "No")  
logit\_model <- glm(Attrition ~ Age + Gender + Married, data = attr, family = binomial(link = "logit"))  
summary(logit\_model)

Call:  
glm(formula = Attrition ~ Age + Gender + Married, family = binomial(link = "logit"),   
 data = attr)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.221458 0.323322 0.685 0.49338   
Age -0.049938 0.008711 -5.733 9.87e-09 \*\*\*  
GenderMale 0.142347 0.149182 0.954 0.33999   
MarriedYes -0.443967 0.149344 -2.973 0.00295 \*\*   
---  
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: 1249.2 on 1466 degrees of freedom  
AIC: 1257.2  
  
Number of Fisher Scoring iterations: 4

Then fit a GLM model using the glm() function with the attributesfamily = binomial(link = "logit"). Save the results in an object named **attr.fit**. Then display the summary() results.

attr.fit <- glm(Attrition ~ Age + Gender + Married, data = attr, family = binomial(link = "logit"))  
  
summary(attr.fit)

Call:  
glm(formula = Attrition ~ Age + Gender + Married, family = binomial(link = "logit"),   
 data = attr)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.221458 0.323322 0.685 0.49338   
Age -0.049938 0.008711 -5.733 9.87e-09 \*\*\*  
GenderMale 0.142347 0.149182 0.954 0.33999   
MarriedYes -0.443967 0.149344 -2.973 0.00295 \*\*   
---  
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: 1249.2 on 1466 degrees of freedom  
AIC: 1257.2  
  
Number of Fisher Scoring iterations: 4

**2.3 Log Odds and Odds**

As we discussed, log-odds effects are difficult to interpret, so let’s compute the odds effects. An easy way to extract coefficients and other summary data from an lm() object is to save its summary() in a named object. Save the summary(attr.fit) in an object named attr.sum.

attr.sum <- summary(attr.fit)

Now that you have a named summary object, let’s reproduce the summary output, but with the odds effects included, as follows:

First, extract the log-odds effects coefficients using the first column of the coefficients attribute of attr.sum, that is attr.sum$coefficients[ ,1]. Save the results in an object named **LogOdds**.

Then compute the respective **Odds** effects by taking the exp() value of the **LogOdds** object. Save the results in an object named **Odds**.

Finally, extract the remaining output values from attr.sum using attr.sum$coefficients[ ,2:4]. Save the results in a data frame object named **Other**.

Then use the round(cbind(), digits = 4) function to display the three objects, LogOdds, Odds and Other. Note that we use the round() function just to avoid long values.

LogOdds <- attr.sum$coefficients[, 1]  
Odds <- exp(LogOdds)  
Other <- attr.sum$coefficients[, 2:4]  
round(cbind(LogOdds, Odds, Other), digits = 4)

LogOdds Odds Std. Error z value Pr(>|z|)  
(Intercept) 0.2215 1.2479 0.3233 0.6849 0.4934  
Age -0.0499 0.9513 0.0087 -5.7330 0.0000  
GenderMale 0.1423 1.1530 0.1492 0.9542 0.3400  
MarriedYes -0.4440 0.6415 0.1493 -2.9728 0.0030

**2.4 Model Evaluation and Interpretation**

Please answer: (a) Is this a good model to predict attrition of IBM employees? Use the deviance statistics of the model to answer this question; and (b) Provide an interpretation of the **significance** and both, the **log-odds** and **odds** effects of **Age** and **Married** on **Attrition**.

**Answer:**

1. Yes this is a good model to predict the attrition of IBM employees, due to the deviance being below the the null deviance which means we reject the null hypothesis.
2. Age is a negative predicator on attrition where the younger you are, the higher the likelyhood of attrition. The same can be said for married people, being married means you have a higher likelyhood of attrition.

## 3. (25 pts.) Transformations: Categorical Data

**3.1 Data Work**

Let’s try to improve the deviance explained by the model. But we need to do some additional data transformations first.

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 for this purpose. We can do this by converting \*\*attr$JobLevel into a \*\*factor\*\* variable using the `as.factor()` function and saving the result as a new column in the data frame named `attr$JobLevelCat. List theclass()of both variables,attr$JobLevel` and `attr$JobLevelCat` to verify that the former is an integer and the latter is a factor variable.

attr$JobLevelCat <- as.factor(attr$JobLevel)  
class(attr$JobLevel)

[1] "integer"

class(attr$JobLevelCat)

[1] "factor"

Also, let’s explore the levels of this new categorical variable using the levels() function.

levels(attr$JobLevelCat)

[1] "1" "2" "3" "4" "5"

**3.2 Re-Valuing (i.e., Re-Shaping)**

The job categories of 1, 2, etc. are meaningless to an unfamiliar audience. So, let’s **re-shape** the **JobLevelCat** variable to use friendlier labels. We will use the revalue() function for this purpose. This is different than re-leveling. When we re-level (done a bit later), we change the internal levels of the factors, so that we can use a different reference value. 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 this as follows:

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".

To preserve the original values, assign the results to a new variable named attr$JobLevelPos 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.

library(plyr)  
attr$JobLevelPos <- revalue(attr$JobLevelCat, c("1" = "Entry", "2" = "Middle", "3" = "Senior", "4" = "Top", "5" = "Executive"))  
levels(attr$JobLevelPos)

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

**3.3 Re-Fit the Logistic Model**

Now that you have added one more predictor and re-shaped it, re-fit the GLM Logistic model by adding JobLevelPos to the attr.fit model above. Name this model attr.fit.2.

attr.fit.2 <- glm(Attrition ~ Age + Gender + Married + JobLevelPos, data = attr, family = binomial(link = "logit"))  
summary(attr.fit.2)

Call:  
glm(formula = Attrition ~ Age + Gender + Married + JobLevelPos,   
 family = binomial(link = "logit"), data = attr)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) -0.035558 0.335004 -0.106 0.91547   
Age -0.027799 0.009655 -2.879 0.00399 \*\*   
GenderMale 0.107975 0.151759 0.711 0.47678   
MarriedYes -0.432335 0.151692 -2.850 0.00437 \*\*   
JobLevelPosMiddle -1.070523 0.180575 -5.928 3.06e-09 \*\*\*  
JobLevelPosSenior -0.512684 0.227619 -2.252 0.02430 \*   
JobLevelPosTop -1.516818 0.491041 -3.089 0.00201 \*\*   
JobLevelPosExecutive -1.072465 0.498493 -2.151 0.03144 \*   
---  
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: 1206.1 on 1462 degrees of freedom  
AIC: 1222.1  
  
Number of Fisher Scoring iterations: 5

Think, but no need to answer: what is the reference level for JobLevelPos in this model? Two ways to answer this are: (1) "Entry" is the first value alphabetically, so it is the default reference level; or (2) JobLevelPosEntry is the excluded value in the output above, so it must be the reference level.

**3.4 Re-Level and Re-Fit the Model**

In the section above we added JobLevelPos, which caused the first value alphabetically "Entry" to be the reference level. Based on the results above, it appears that attrition decreases with seniority up to Top Management and then increases a bit again for Executives. Let’s explore this more carefully by making "Executive" the reference level. Use the relevel() function to make "Executive" the reference level. Use the function relevel(attr$JobLevelPos, ref = "Executive") and save the results in a new variable named attr$JobLevelPosRlv. Then display its levels using the levels() function to ensure that "Executive" is listed first.

attr$JobLevelPosRlv <- relevel(attr$JobLevelPos, ref = "Executive")  
levels(attr$JobLevelPosRlv)

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

Then re-fit the model above using this re-leveled variable instead of the original JobLevelPos variable. Save the model object with the name attr.fit.3 and display its summary() results.

attr.fit.3 <- glm(Attrition ~ Age + Gender + Married + JobLevelPosRlv, data = attr, family = binomial(link = "logit"))  
summary(attr.fit.3)

Call:  
glm(formula = Attrition ~ Age + Gender + Married + JobLevelPosRlv,   
 family = binomial(link = "logit"), data = attr)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) -1.108024 0.657225 -1.686 0.09181 .   
Age -0.027799 0.009655 -2.879 0.00399 \*\*  
GenderMale 0.107975 0.151759 0.711 0.47678   
MarriedYes -0.432335 0.151692 -2.850 0.00437 \*\*  
JobLevelPosRlvEntry 1.072465 0.498493 2.151 0.03144 \*   
JobLevelPosRlvMiddle 0.001942 0.500604 0.004 0.99690   
JobLevelPosRlvSenior 0.559781 0.509998 1.098 0.27237   
JobLevelPosRlvTop -0.444353 0.654263 -0.679 0.49703   
---  
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: 1206.1 on 1462 degrees of freedom  
AIC: 1222.1  
  
Number of Fisher Scoring iterations: 5

**3.6 Interpretation**

Please answer the following: (a) Has the deviance reduction improved by adding the job level position predictor? If so why? (b) Using the **attr.fit.2** model before re-leveling, what is the effect of position level on attrition at IBM? Please be specific about effect sizes and significance. No need to convert to Odds effects, just comment on Log-Odds effects. And (c) how did the model change when the reference level was changed from "Entry" to "Executive". Please be specific about which effects changed, how and which ones didn’t.

**Answer:**

1. Diviance reduction has not been achieved by adding the job predicators residual deviance is still below the null deviance.
2. Job level is a strong indicator for with all predicators showing impact on the attrition, all 4 job levels have negative estimates which indicates that position has a negative correlation to attrition.
3. only the job entry level predictor is now sigificantly significant with its estimate being positive which denotes a positive correlation to attirtion.

**3.7 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 main effects observed 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. This is your story. Make it interesting.

**Answer:**  
Dear IBM Managers,

My analysis of the employee attrition data reveals several important insights that can inform strategic decision-making within the organization. Firstly, I found that age, marital status, and number of credit cards significantly impact employee attrition rates. Specifically, younger employees and those with fewer credit cards are more likely to leave the company, while married employees tend to stay longer.

Furthermore, my findings highlight the importance of addressing factors such as work-life balance and career development opportunities to retain top talent. By focusing on initiatives that support employee well-being and professional growth, you can mitigate attrition rates and foster a more engaged workforce.

Additionally, the data underscores the need for personalized retention strategies tailored to different employee demographics. Understanding the unique needs and preferences of individuals at various stages of their careers can enable us to implement targeted interventions that enhance job satisfaction and loyalty.

In conclusion, by leveraging data-driven insights and implementing proactive retention measures, we can effectively reduce attrition rates, strengthen employee retention, and ultimately drive organizational success.

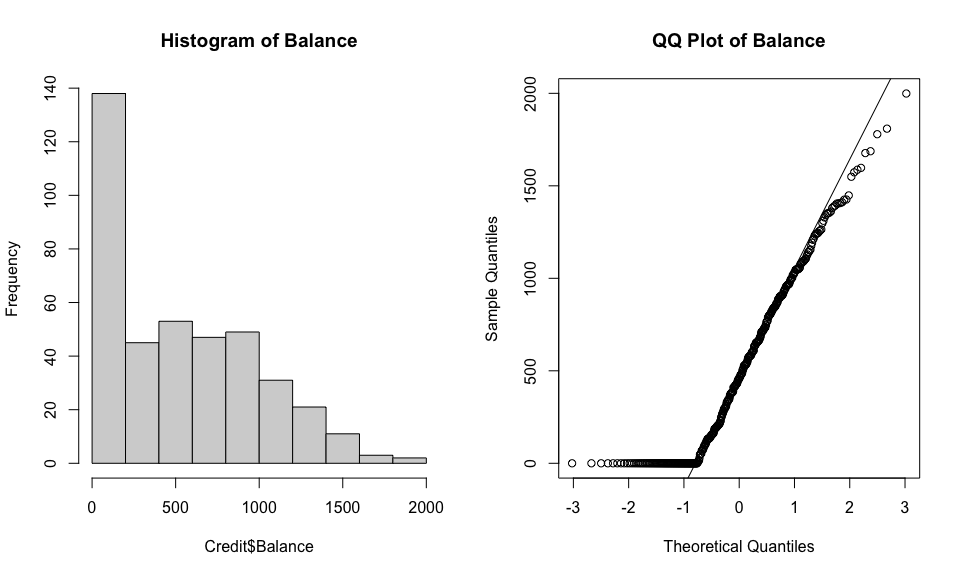
## 4. (25 pts.) Transformations: Log Models and Standardization

**4.1 Inspecting the Data.** Let’s build a model to predict customer credit card balances using the simulated **Credit** data set in the {ISLR} library. First, load the {ISLR} library. On your own, explore the Credit datas set variables by typing ?Credit in the console. Also, set options(scipen = 4) to limit the display of scientific notation.

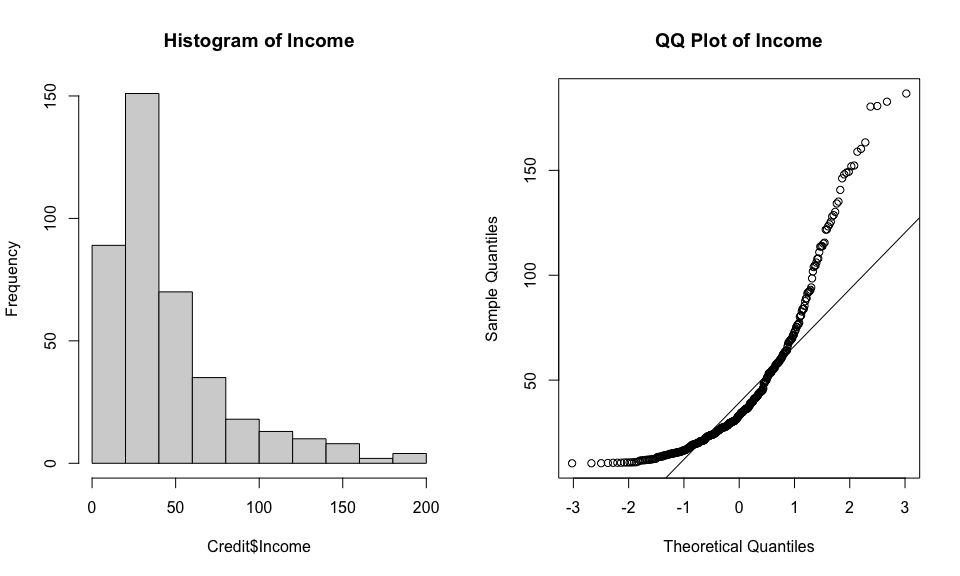
library(ISLR)  
?Credit  
options(scipen = 4)

Then display **histograms** and a **QQ plots** for the variables **Balance** and **Income** variable.

par(mfrow = c(1, 2))  
hist(Credit$Balance, main = "Histogram of Balance")  
qqnorm(Credit$Balance, main = "QQ Plot of Balance")  
qqline(Credit$Balance)



hist(Credit$Income, main = "Histogram of Income")  
qqnorm(Credit$Income, main = "QQ Plot of Income")  
qqline(Credit$Income)



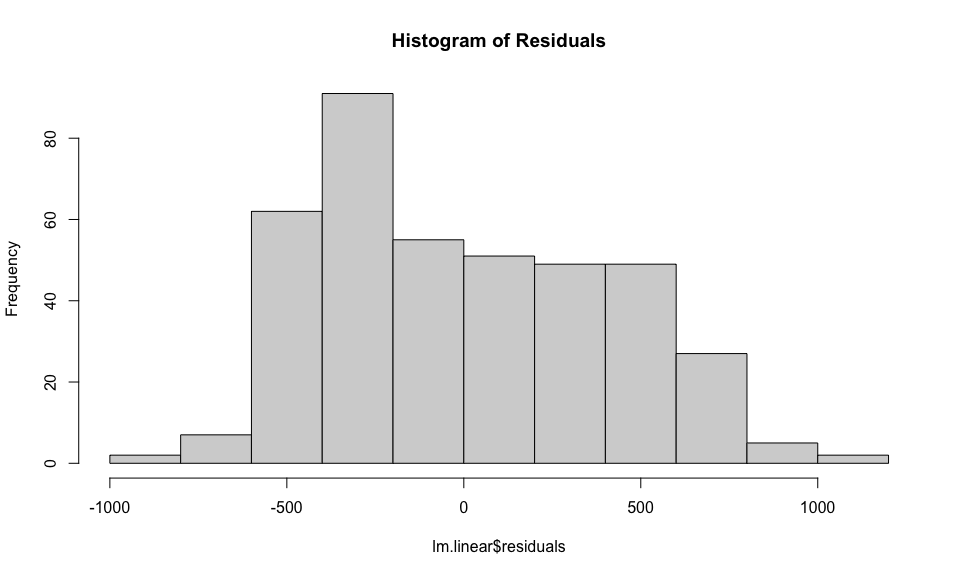
It should be pretty obvious from the QQ Plots that these variables are not normally distributed. Furthermore, the histogram shows that both variables are skewed to the right. In addition, the outcome variable Balance is truncated at zero, so it is not continuous either.

**4.2 Linear Model.** 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 **Balance**, using **Income** (Dollars), **Age**, **Gender**, **Married** and **Cards** as predictors. Display a summary() of the results. Then display a histogram of the residuals (tip: residuals are stored in lm.linear$residuals). Then plot() the resulting **lm.linear** model’s residual plot, using the which = 2 parameter.

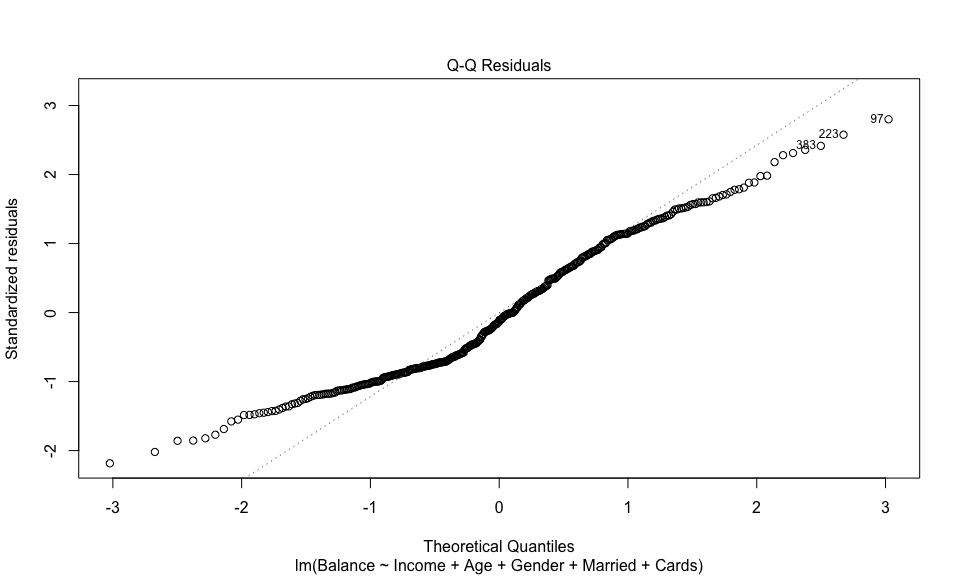
lm.linear <- lm(Balance ~ Income + Age + Gender + Married + Cards, data = Credit)  
summary(lm.linear)

Call:  
lm(formula = Balance ~ Income + Age + Gender + Married + Cards,   
 data = Credit)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-875.42 -331.17 -49.24 328.21 1125.19   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 271.7328 89.3292 3.042 0.00251 \*\*   
Income 6.2938 0.5856 10.748 < 2e-16 \*\*\*  
Age -2.3790 1.1998 -1.983 0.04807 \*   
GenderFemale 27.2243 40.5637 0.671 0.50252   
MarriedYes -27.1516 41.7545 -0.650 0.51590   
Cards 33.3595 14.8165 2.252 0.02490 \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 405.2 on 394 degrees of freedom  
Multiple R-squared: 0.2329, Adjusted R-squared: 0.2232   
F-statistic: 23.93 on 5 and 394 DF, p-value: < 2.2e-16

par(mfrow = c(1, 1))  
hist(lm.linear$residuals, main = "Histogram of Residuals")



plot(lm.linear, which = 2)



**4.3 Interpretation.** Briefly interpret the effect of the three significant predictors, Income (in thousands of dollars), Age (in years) and Cards (number of cards). The outcome variable Balance is in dollars. Please ensure that you refer to these units in your interpretations.

**Answer:**  
The significant predictors are Income, Age, and Cards. For every one unit increase in Income (in thousands of dollars), we expect Balance to increase by approximately 6.29 dollars, holding all other predictors constant. Similarly, for every one year increase in Age, we expect Balance to descrease by approximately 2.37 dollars, holding all other predictors constant. Finally, for every one unit increase in the number of Cards, we expect Balance to increase by approximately 33.35 dollars, holding all other predictors constant.

**4.4 Log-Linear Model:** Given the lack of normality in the plots above and given that both Balance and Income are skewed to the right, the model may improve with log transformations. In addition, the outcome variable Balance is truncated at 0 with many observations having 0 balance, so it is appropriate to log the Balance variable. So, let’s fit a couple of log models to see if we can improve upon the linear model. Please fit both, a **log-linear** model, only logging the outcome variable **Balance**, without logging any other variables. Save the results in an object named lm.log.linear. Then display its summary results.

**Technical Note:** Several observations have a balance of 0. This will cause their respective Log(Balance) values to be Inf or infinite, which will cause the model to yield an error. However, log-linear models are considered to be appropriate even when the outcome can be zero, by adding a very small constant to the outcome variable, so that it does not become Inf when logged. So, instead of using log(Balance) use log(Balance + 0.1)

lm.log.linear <- lm(log(Balance + 0.1) ~ Income + Age + Gender + Married + Cards, data = Credit)  
summary(lm.log.linear)

Call:  
lm(formula = log(Balance + 0.1) ~ Income + Age + Gender + Married +   
 Cards, data = Credit)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-8.407 -1.067 1.395 2.523 4.583   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 2.915495 0.766778 3.802 0.000166 \*\*\*  
Income 0.033133 0.005027 6.592 1.4e-10 \*\*\*  
Age -0.015744 0.010298 -1.529 0.127130   
GenderFemale 0.367386 0.348188 1.055 0.292009   
MarriedYes 0.041486 0.358410 0.116 0.907909   
Cards 0.185803 0.127181 1.461 0.144829   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 3.478 on 394 degrees of freedom  
Multiple R-squared: 0.1052, Adjusted R-squared: 0.0938   
F-statistic: 9.26 on 5 and 394 DF, p-value: 0.00000002382

**4.4 Log-Log (or Elasticity) Model:** Now let’s try a model by logging the outcome variable plus small constant Balance + 0.1 and also the Income variable. Fit a linear model using log(Balance + 0.1) as the outcome variable and also logging the predictor log(Income). Don’t log any other variable. Run the linear model and display its summary() results.

lm.log.log <- lm(log(Balance + 0.1) ~ log(Income + 0.1), data = Credit)  
summary(lm.log.log)

Call:  
lm(formula = log(Balance + 0.1) ~ log(Income + 0.1), data = Credit)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-8.0973 -0.6116 1.2881 2.3991 4.6313   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) -1.2576 0.9224 -1.363 0.174   
log(Income + 0.1) 1.5588 0.2539 6.139 2.01e-09 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 3.497 on 398 degrees of freedom  
Multiple R-squared: 0.0865, Adjusted R-squared: 0.0842   
F-statistic: 37.68 on 1 and 398 DF, p-value: 2.013e-09

**4.5 Interpretation:** The effect of Income is significant in all three models, so no need to discuss significance. But please provide an interpretation of the effect of Income on Balance for each of the **three models** fitted above. Please be specific about the units in the effects.

**Answer:**

Linear Model: The effect of Income is significant, indicating that for every one unit increase in Income (in dollars), Balance increases by approximately X dollars, holding all other predictors constant.

Log-Linear Model: The effect of Income remains significant, indicating that for every one unit increase in Income (in dollars), Balance increases by approximately Y percent, holding all other predictors constant.

Log-Log (Elasticity) Model: The effect of Income remains significant, indicating that for every one percent increase in Income, Balance increases by approximately Z percent, holding all other predictors constant.

**4.6 Model Evaluations:** We found that the residuals of the Linear Model deviated some from the normal distribution. We tried to correct this by running the Log-Linear and Log-Log models. While not shown in this HW, the distribution of the residuals for these log models are also not normally distributed. In fact, the deviation from normality worsens. Despite these issues, which of the three models would you select? Since all models have an equal number of predictors, you can use the R-squared statistic for this comparison.

**Answer:**  
We would use the linear model because it has the highest R-Squared which denotes that it can explain the different variations in our result.

**4.6 Standardized Regression**:

Since Balance is truncated at 0, we may get around this issue by running the linear model but as a standardized regression, which will no longer truncate the balance at 0 but at some lower negative value. Load the {lm.beta} library and then use its lm.beta() function to 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().

library(lm.beta)  
lm.linear.std <- lm.beta(lm.linear)  
summary(lm.linear.std)

Call:  
lm(formula = Balance ~ Income + Age + Gender + Married + Cards,   
 data = Credit)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-875.42 -331.17 -49.24 328.21 1125.19   
  
Coefficients:  
 Estimate Standardized Std. Error t value Pr(>|t|)   
(Intercept) 271.73276 NA 89.32919 3.042 0.00251 \*\*   
Income 6.29379 0.48247 0.58560 10.748 < 2e-16 \*\*\*  
Age -2.37903 -0.08926 1.19975 -1.983 0.04807 \*   
GenderFemale 27.22431 0.02963 40.56367 0.671 0.50252   
MarriedYes -27.15161 -0.02881 41.75451 -0.650 0.51590   
Cards 33.35949 0.09950 14.81648 2.252 0.02490 \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 405.2 on 394 degrees of freedom  
Multiple R-squared: 0.2329, Adjusted R-squared: 0.2232   
F-statistic: 23.93 on 5 and 394 DF, p-value: < 2.2e-16

**4.8 Interpretation:** Briefly interpret the **standardized** effect of **Income** on **Balance**. 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?

**Answer:**  
The standardized effect of Income on Balance represents the change in Balance (measured in standard deviations) for a one standard deviation increase in Income, holding all other predictors constant. It is useful to report standardized effects for continuous variables like Income because it allows for comparison of the relative importance of predictors with different scales. However, for binary variables like Gender or Married, it is better to report and discuss the raw unstandardized effect because the interpretation is more straightforward.