Regression (Bivariate and Multiple) Homework

replace with partner names

replace with date

Part A

General Questions

1. **QUESTION:** Describe the purpose of using bivariate-linear regression. Include in your answer the type of data needed for linear regression analyses.

ANSWER: Bivariate regression is used to understand the linear relationship between two numeric variables (interval or ratio). The relationship allows for using one variable to predict the outcome of another variable especially when information is unknown about the criterion variable.

2. **QUESTION:** Identify the two coefficient values that are needed for a regression analysis and describe what they are.

ANSWER: b0 - y-intercept; the value of the criterion variable (y) when the predictor variable (x) is 0. b1 - the regression coefficient, or slope that represents the unit change (increase or decrease) in the criterion for each unit change in the predictor.

3. **QUESTION:** The regression model is compared to another model, a simpler one. Identify what that model is and describe what the error in that model represents.

ANSWER: A mean-based model is a simple model that describes all of the error in the criterion variable. In absence of knowing information about a second (predictor) variable, the mean is the best guess or estimate of a score in a sample. The error associated with the mean-based model, in terms of SS, is referred to SStotal.

4. **QUESTION:** If the SStotal is 2000 and the SSresidual are 1000. What is the proportion of variance in the criterion is accounted for by the predictor?

ANSWER: SSmodel = SStotal - SSresidul; SSmodel/SStotal = .50

Part B

Before you begin

This homework exercise involves having you answer some questions, write some code, and create a nice HTML file with your results. When asked different questions, simply either type your coded or written responses after the ANSWER message. When asked to write code to complete sections, type your code

in the empty code blocks that follow the ANSWER message (between the back ticks). After adding that code, you must make sure that it will execute. So remember to read the content and run each line of your code as you write it so that you know it executes correctly. If your code does not execute, then RMarkdown won't know what you are telling it to do and your HTML file will not be produced. Also, don't create your HTML file until you finish and know that all of your code works correctly.

Besides lecture notes, some videos help with understanding R output. I recommend watching: - https://www.youtube.com/watch?v=eTZ4VUZHzxw (https://www.youtube.com/watch?v=eTZ4VUZHzxw) - https://www.youtube.com/watch?v=q1RD5ECsSB0 (https://www.youtube.com/watch?v=q1RD5ECsSB0)

1. Installing and using libraries in RStudio

1.1. Use the RStudio interface to install packages/libraries. Go to the Tools option and select Install Packages. Type the package name(s) correctly using the proper letter casing. Also, make sure that you *check the box to Install Dependencies*. Do not install with code.

- car
- lattice
- QuantPsyc
- Mac users, most of you should already have XQuartz installed from the first R class, but please make sure that you have it downloaded and installed in your Applications folder. See the original download instruction file if you need help.
- Did you still have problems with the ncvTest() or durbinWatsonTest() functions from the car library from the last exercise? If so, the errors did not resolve themselves by installing a new version of R (3.2.3), so do the following. Remove the # from the code block below and execute the code. After doing so, add back the # so the code won't execute when you knit your file. Please let me know if this solved your error with the ncvTest() and/or durbinWatsonTest() functions.

1.2. New functions used for this assignment:

- durbinWatsonTest() for testing assumptions; car library
- lm() for evaluating linear models; built-in library
- lm.beta() for calculating standarized regression coefficients; QuantPsyc library
- ncvTest() for testing assumptions; car library
- par() for changing graphing parameters; built-in library
- plot() for examining lm regression plots; built-in library
- round() for rounding values; built-in library
- summary() for obtaining a summary of the model components
- shapiro.test() for testing normality; built-in stats library
- sqrt() for transforming data; built-in library

2. Loading libraries

Use the library() function to load the following libraries: car, lattice, QuantPsyc

ANSWER:

```
library(car)
library(lattice)
library(QuantPsyc)
```

3. Checking your working directory

Always make sure that your working directory points to Psyc109 on your desktop.

```
getwd()
```

```
## [1] "X:/Progs/_install/R/Exercises/Homework"
```

```
#setwd("C:/users/gcook/desktop/Psyc109/")
```

Part B

1. General Linear Regression

1.1. Linear regression is an approach for modeling the linear relationship between a criterion y and one or more predictor/explanatory variables (or independent variables).

1.2. There are two common forms of *linear regression*:

1.2.1. *Simple-linear regression* is used when the goal is to use one predictor variable in the linear regression equation to predict a criterion variable (Ex: formula: criterion ~ predictor). This will produce a linear model of your criterion as-a-function-of your predictor.

1.2.2. *Multiple-linear regression* is used to indicate there is more than one predictor in the linear regression equation (Ex: formula: criterion ~ predictor1 + predictor2 + predictor3).

Both of these forms of regression analysis have assumptions that should be met in order to interpret the data. The assumptions are addressed in sections below.

2. Simple (bivariate) Linear Regression

2.1. Understanding Simple Bivariate Linear Regression.

For a simple-linear regression there is:

- One predictor in the equation
- An unstandardized regression coefficient that represents the marginal relationship between the criterion and the predictor variable
- The standardized regression coefficient equals is the correlation value

2.2. Reading in the data

One of the first steps in completing a simple linear regression is to plot your data on a scatter plot. For that we are going to need to bring some data into our workspace. We will read in a data file named "Album Sales.csv" and assign it to a data frame object named ALBUM. Examine the structure once you are done so that you know your variables, their scale of measurement, and their format.

```
ALBUM <- read.csv("Album Sales.csv") # Read in csv file and assign as data frame object
str(ALBUM)
```

```
## 'data.frame':    200 obs. of  5 variables:
##  $ Id     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Sales  : int  330 120 360 270 220 170 70 210 200 300 ...
##  $ Ads    : num  10.3 985.7 1445.6 1188.2 574.5 ...
##  $ Airplay: int  43 28 35 33 44 19 20 22 21 40 ...
##  $ Attract: int  10 7 7 7 5 5 1 9 7 7 ...
```

As usual, the first thing to do with any new data set is to view the data frame. You will notice that the str() of the data frame indicates that there are 200 observations of data (rows) and 3 variables (columns). The varibles/columns are:

- *Id* represents the unique Id for a rock band album
- *Sales* represents album sales (in thousands of pounds)
- *Ads* represents the amount of money spent on the advertising budget (in thousands of pounds)
- *Airplay* represents the number of times an album is played on the radio during the week prior to release
- *Attract* represents the attractiveness of the band members [on a scale from 0 (not attractive at all to 10 (most attractive as possible)]

```
View(ALBUM)
```

3. Producing a correlation matrix

3.1. As we have done before, you can specify all of the variable pairs you want to correlate, but if there are multiple pairs, there is an easier way than doing this repeatedly for each x-y pair. You can simply use the cor() function on the ALBUM object itself (e.g., cor(ALBUM)) rather than specify the variable names as you have done before (e.g., cor(ALBUM$sales$, $ALBUM$Ads)).

```
cor(ALBUM)
```

```
##                  Id       Sales         Ads     Airplay    Attract
## Id       1.00000000 -0.09197545 -0.07663863 -0.08014404 0.04965499
## Sales   -0.09197545  1.00000000  0.57848774  0.59891880 0.32611105
## Ads     -0.07663863  0.57848774  1.00000000  0.10188281 0.08075151
## Airplay -0.08014404  0.59891880  0.10188281  1.00000000 0.18198863
## Attract  0.04965499  0.32611105  0.08075151  0.18198863 1.00000000
```

3.2. However, the decimal points in the correlation matrix are a little busy. We can use the built-in round() function to round the results by specifying the number of decimals to round to. With R, you can nest functions inside functions, so we will nest cor() inside round() in order to round the correlation values for all the variable pairs. Remember, cor() using Pearson's r by default.

```
round(cor(ALBUM), 3)  # rounding makes it easier to look at
```

```
##             Id  Sales    Ads Airplay Attract
## Id       1.000 -0.092 -0.077  -0.080   0.050
## Sales   -0.092  1.000  0.578   0.599   0.326
## Ads     -0.077  0.578  1.000   0.102   0.081
## Airplay -0.080  0.599  0.102   1.000   0.182
## Attract  0.050  0.326  0.081   0.182   1.000
```

```
round(cor(ALBUM[, -1]), 3)  # because ID is meaningless for our correlations, we can remo
ve column 1 from the ALBUM object because ID is the first column in the ALBUM data frame
```

```
##         Sales   Ads Airplay Attract
## Sales   1.000 0.578   0.599   0.326
## Ads     0.578 1.000   0.102   0.081
## Airplay 0.599 0.102   1.000   0.182
## Attract 0.326 0.081   0.182   1.000
```

4. Define a simple linear model

4.1. We will define a linear model using the built-in lm() function by specifying arguments of the function. The general form of the regression model is as follows:

Regression model: Y-hat = b0 + b1*X

Or in R, this is:

mymodel <- lm(formula = criterion ~ predictor, data = mydataframe, na.action = some action)*

- *mymodel* is an object that contains information about the linear model; summary() will be useful for inspecting the model
- *criterion* is the predicted variable
- *predictor(s)* are the variable used to predict scores on the criterion
- *data* is the name of the data frame object
- *na.action* (optional) allows you to specify a complete data set if you wish to drop out anyone with missing values; this in an alternative to subetting when using regression models

4.2. You can specify the predictor and the criterion, "y as a function of x", as well as the data frame.

```
# Using the dataframe$variable approach
ALBUM.model <- lm(formula = ALBUM$Sales ~ ALBUM$Ads)

# Specifying the data object makes things easier
ALBUM.model <- lm(formula = Sales ~ Ads, data = ALBUM)

# If there are missing values, we can remove them (good idea to add this)
ALBUM.model <- lm(formula = Sales ~ Ads, data = ALBUM, na.action = na.exclude)
```

5. Examining and interpreting the simple bivariate linear model

5.1. If you only wanted to examine the y-intercept (b0) and the regression coefficient (b1), you can simply ask to see them by calling the model object.

```
ALBUM.model  # Displays only the coefficients from the model
```

```
##
## Call:
## lm(formula = Sales ~ Ads, data = ALBUM, na.action = na.exclude)
##
## Coefficients:
## (Intercept)          Ads
##   134.13994      0.09612
```

Or you can summarize other details of the linear model by using the built-in summary() function.

```
summary(ALBUM.model)  # provides other model elements
```

```
##
## Call:
## lm(formula = Sales ~ Ads, data = ALBUM, na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -152.949  -43.796   -0.393   37.040  211.866
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.341e+02  7.537e+00  17.799   <2e-16 ***
## Ads         9.612e-02  9.632e-03   9.979   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65.99 on 198 degrees of freedom
## Multiple R-squared:  0.3346, Adjusted R-squared:  0.3313
## F-statistic: 99.59 on 1 and 198 DF,  p-value: < 2.2e-16
```

5.2. Interpreting Overall Model Fit

5.2.1. The *Multiple R-squared* represents the proportion of variance in the criterion that is accounter for by the predictor. The square root of this is Pearson's correlation coefficient. Because there is only one predictor, this multiple R-squared is really just r squared. Because the estimates are based on samples, they are not perfect for making inferences about the populations you really care about. The adjusted R-squared is adjusted for shrinkage, or loss of predictive power that occurs when using sample data to predict population data; the more error in the regression model (e.g., residuals), the more the adjusted R-squared will differ from the unadjusted R-squared.

5.2.2. The *F-statistic* represents the ANOVA test value (F value) for the ratio test of the model. A statistically significant result indicates that the regression model explains the data better than model based on the mean of album sales (mean-based model). People often compare the *p*-value to alpha to decide if the regression model fits the data better than the mean-based model. This ANOVA test only tells you whether the overall model fits the data; it does not examine the components of the model (e.g., b0, b1, etc.).

5.3. Interpreting Specific Model Parameters (coefficents part of the output table)

The coeffiencts report is displayed in 2 rows and 4 columns.

5.3.1. The *Estimate* column displays the two estimated coeffient values y-intercept (b0) and the regression coefficient (b1). The *Std. Error* column displays the error in estimating the coefficients. This error indicates the amount of error in coefficients; small error is good and would indicate that b0 and b1 would not vary as much from sample to sample. The *t-value* column displays the value of the *t*-statistic, which simply tests whether the coeffienct differs from 0 (H0: t = 0), which can be inferred also from examining the *Pr(>|t|)* column which provides the *p*-value for the *t*-test.

5.3.1.1 The first row corresponds to the y-intercept (b0). The y-intercept predicts the number of sales when NO money (e.g., X = 0) is spent on advertisements. A *p*-value less than or equal to alpha may indicate that the y-intercept differs from 0. Mathematically, b0 can be less than 0.

5.3.1.2. The second row corresponds to the regression coefficient/slope (b1). This represents the increase or decrease (depending on whether the correlation is positive or negative) in the number of album sales for each unit change in money spent on advertising. A *p*-value equal to or less than alpha means the slope differs from 0. When there are multiple predictor variables, there will be slopes corresponding the each predictor and the criterion.

6. Examine Model Assumptions

6.1. Model assumptions can and should be inspected both visually and statistically. Passing the model to the built-in plot() function will return a set of plots for inspecting the model assumptions. Other specialized functions from other libraries can also help inspect a model.

Assumptions and things to look for:

6.1.1. The predictor(s) must be either quantitative or categorical (2 categories); the criterion must be quantitative and continuous. The criterion should also be unbounded or unconstrained; for example if a scale that ranges from 1 to 9 is used to measure the criterion and if responses only fall between say 3 and 7, the data are constrained (bounded). An obvious solution is to check the range for the criterion using the built-in range() function.

```
range(ALBUM$Sales)
```

```
## [1]  10 360
```

6.1.2. Predictors should not be restricted and not have variances that are near 0. Range restriction in general is often problematic with regression (leading to attenuated correlations and reduce predictive ability). We can check variance. It's often good to determine if your data range is similar to previous research. If you have a much smaller range, you should investigate reasons why. Otherwise, your correlations and regression coefficients may not map on well with previous research.

```
mean(ALBUM$Ads, na.rm = TRUE)
```

```
## [1] 614.4123
```

```
range(ALBUM$Ads)
```

```
## [1]    9.104 2271.860
```

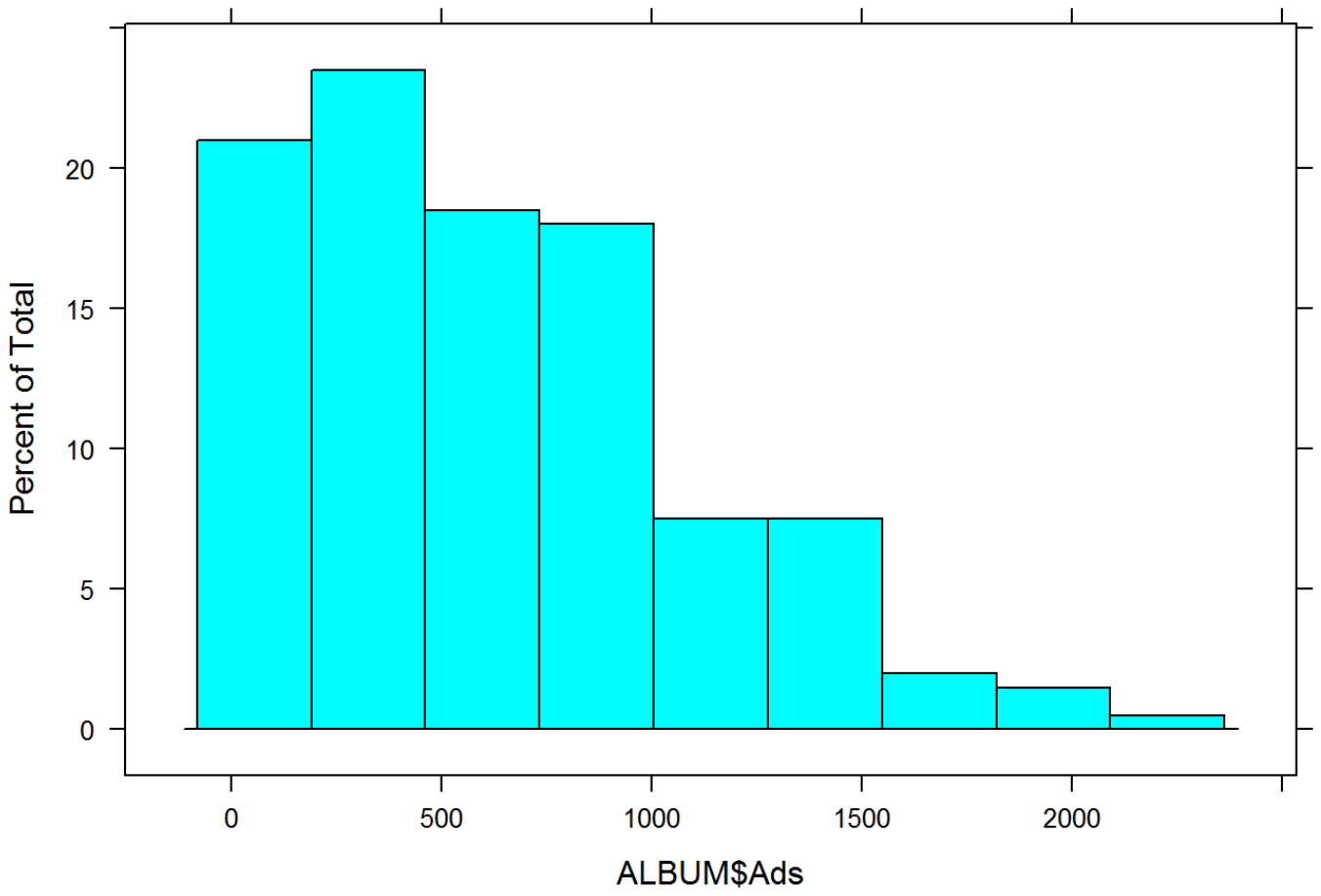
```
var(ALBUM$Ads, na.rm = TRUE) # does not look near 0
```

```
## [1] 235861
```

```
sd(ALBUM$Ads, na.rm = TRUE)  # does not look near 0, so we have variability
```

```
## [1] 485.6552
```

```
#library(lattice)  # we need this for histogram()
histogram(~ ALBUM$Ads)  # notice that the predictor is skewed positively
```



The predictor (advertisements) appears to be skewed positively. We can test whether the shape statistically differs from a normal distribution by using the Shapiro-Wilk test. The shapiro.test() function is part of the built-in stats library so you don't need to load anything special. The D'Agostino test will test for skewness only, not normality, but is part of the moments library so you would need to load it if you wanted to test skewness specifically. The Shapiro-Wilk test value is W (kind of like the z-test value is Z). In order to determine if the distribution is not normal in shape, people often compare the *p*-value that corresponds to Shapiro-Wilk test value (W) to your desired alpha. In this example, we can see that our distribution of money spent on advertisements is not normal if alpha = .05.

```
shapiro.test(ALBUM$Ads)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ALBUM$Ads
## W = 0.92542, p-value = 1.482e-08
```

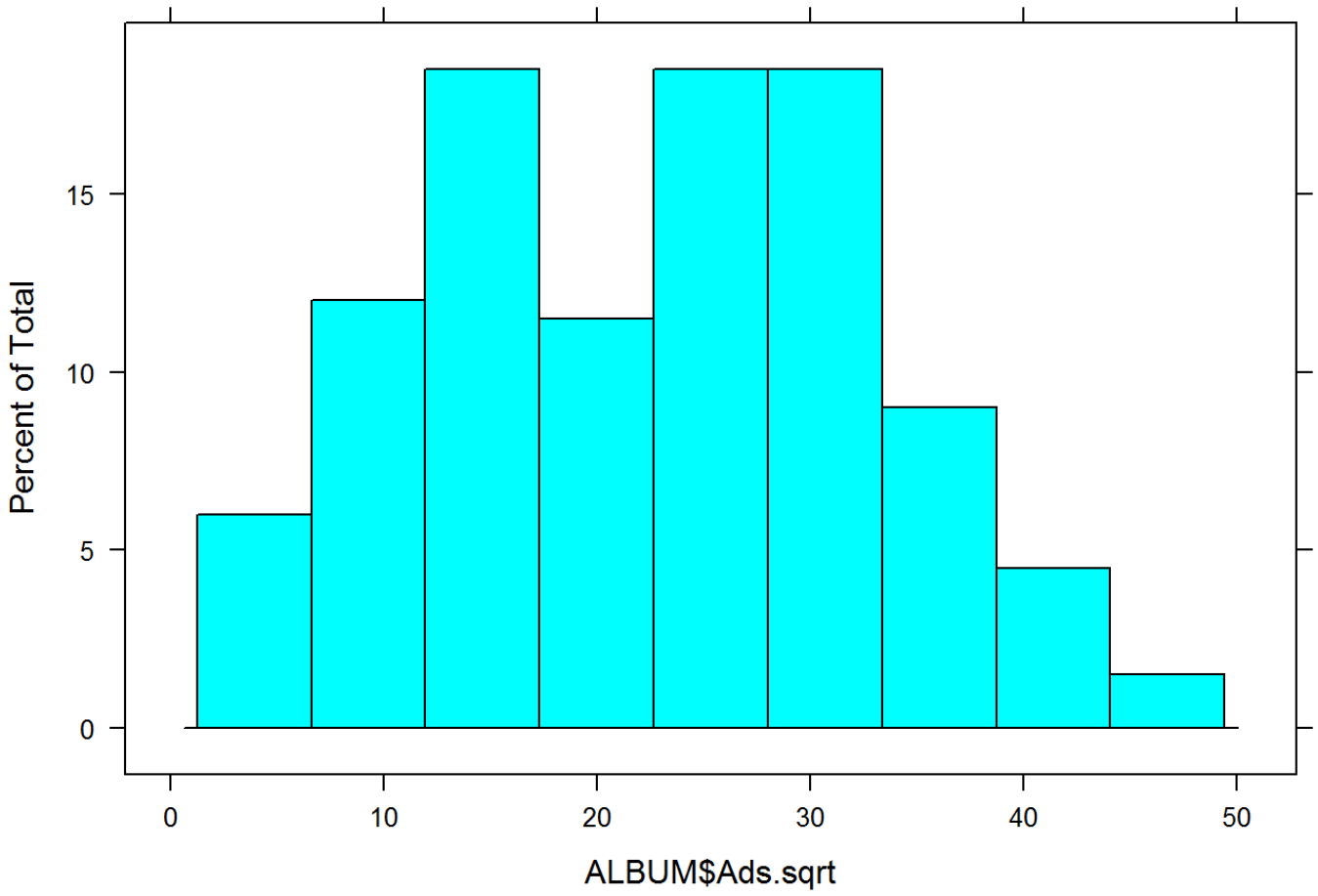
When you have skewness, you should deal address that problem. You can mathematically transform your data from raw values to another metric. However, you need to be careful when interpreting statistics using transformed data because you are no longer dealing with actual raw data. Although we will not deal with transformations much in this class, you can find some discussions online, for example at https://en.wikipedia.org/wiki/Power_transform (https://en.wikipedia.org/wiki/Power_transform) and http://pareonline.net/getvn.asp?v=8&n=6 (http://pareonline.net/getvn.asp?v=8&n=6)

For example, we can create a new variable in our data frame that represents the square root of the Ads variable. We will use the built-in sqrt() function. There are other transformations that can fix your data if this one doesn't work. We won't get into all of them here.

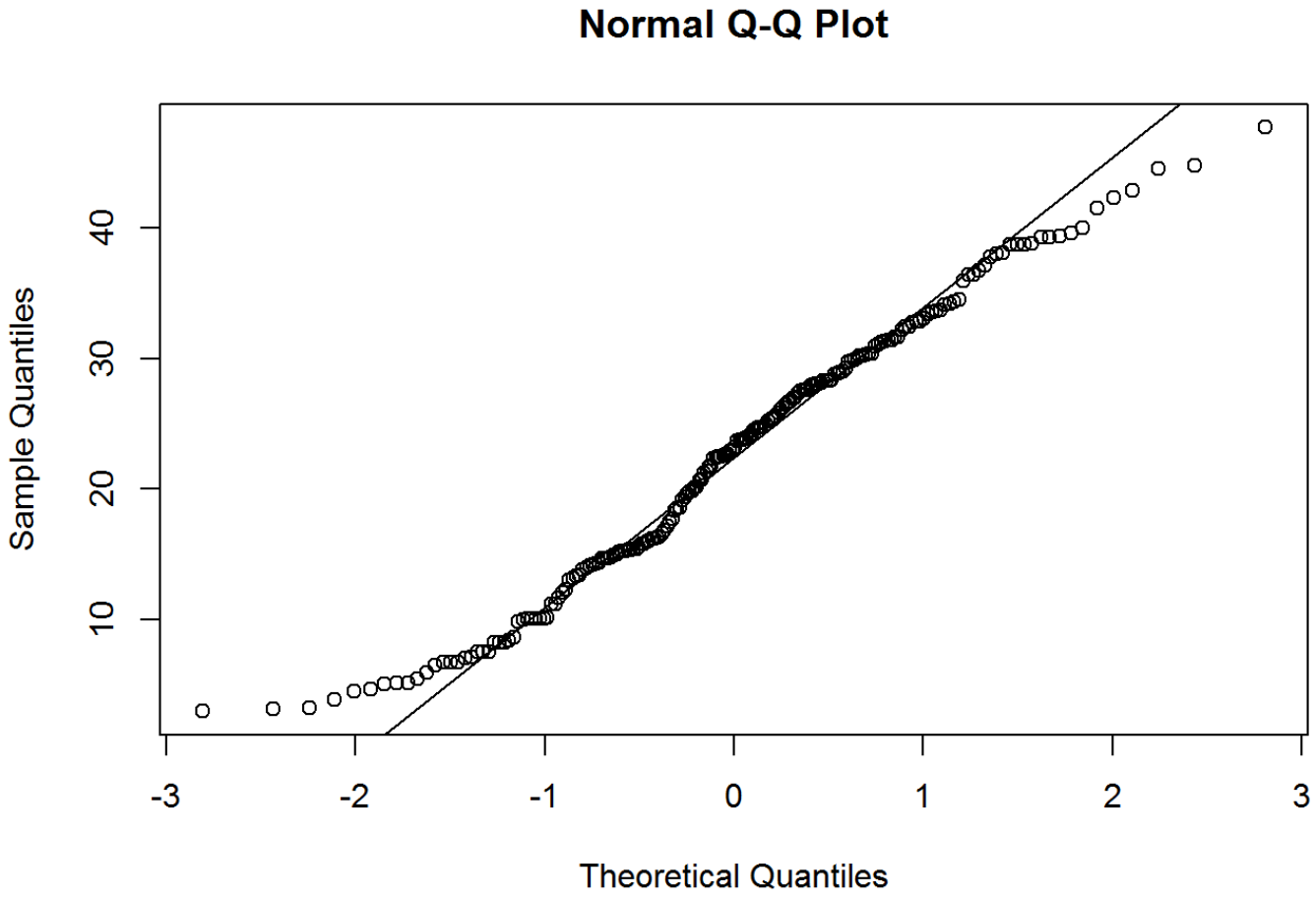
```
ALBUM$Ads.sqrt <- sqrt(ALBUM$Ads)
str(ALBUM)  # check the structure to make sure the variable is there.
```

```
## 'data.frame':    200 obs. of  6 variables:
##  $ Id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Sales   : int  330 120 360 270 220 170 70 210 200 300 ...
##  $ Ads     : num  10.3 985.7 1445.6 1188.2 574.5 ...
##  $ Airplay : int  43 28 35 33 44 19 20 22 21 40 ...
##  $ Attract : int  10 7 7 7 5 5 1 9 7 7 ...
##  $ Ads.sqrt: num  3.2 31.4 38 34.5 24 ...
```

```
histogram(~ ALBUM$Ads.sqrt)  # looks better than before
```




```
qqnorm(ALBUM$Ads.sqrt); qqline(ALBUM$Ads.sqrt)  # the tails of the distribution don't hug the identity line; could be a problem.
```

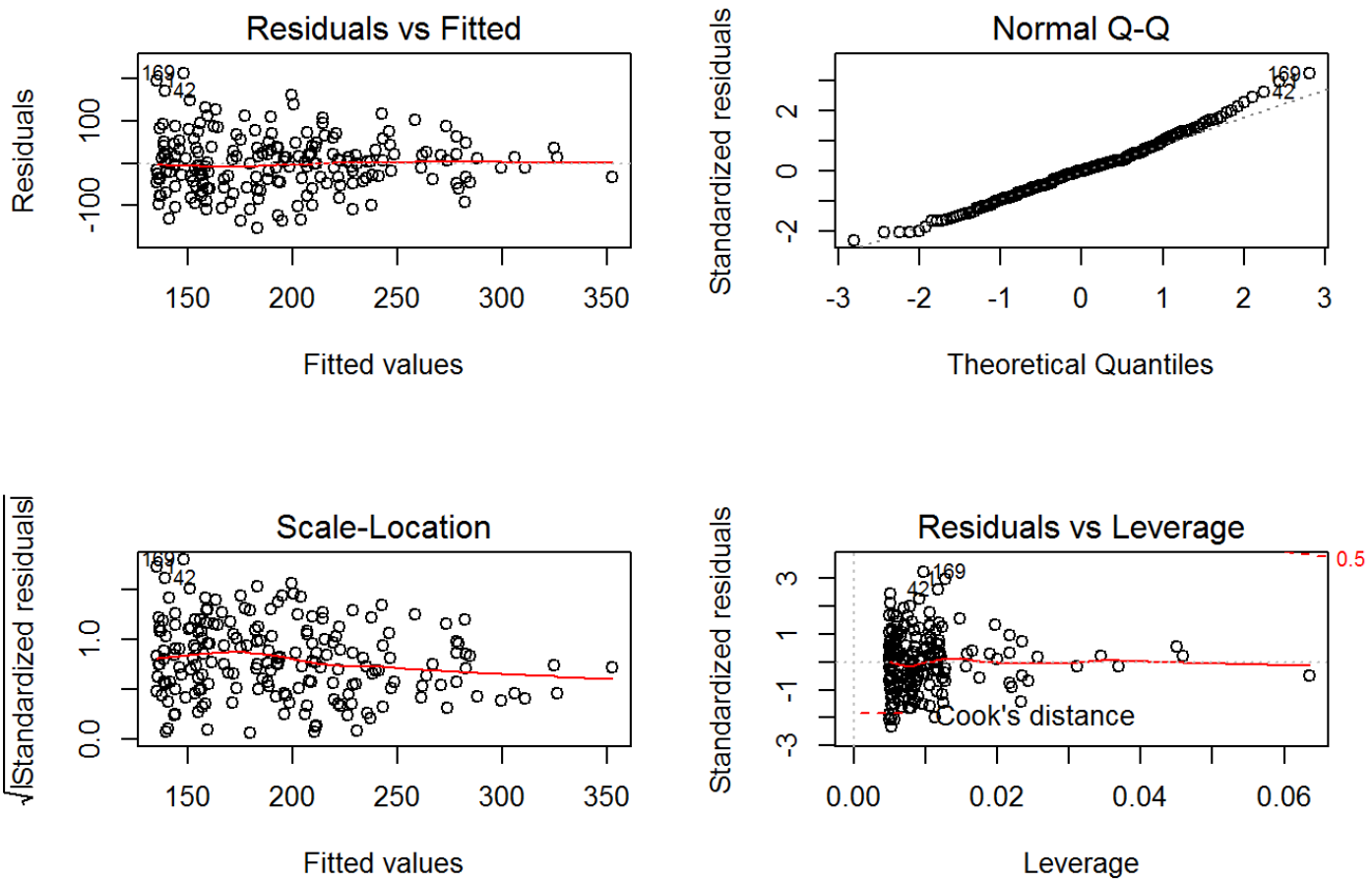



```
shapiro.test(ALBUM$Ads.sqrt) # but these transformed data still have a distribution diffe
rent from normal, see p-value
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ALBUM$Ads.sqrt
## W = 0.98022, p-value = 0.006389
```

6.1.3. Using the plot() function on the model object that you created will produce model plots for examining the assumptions of the model. You can view each plot separately or produce a 2x2 chart that displays them together. However, grouping the plots will reduce their size for visual inspection.

```
par(mfrow = c(2,2))  # adjust the plots in order to plot the 4 graphs on one chart with 2
rows, 2 columns.
plot(ALBUM.model)
```




```
par(mfrow = c(1,1))  # return to 1 row, 1 column
```

6.1.4. The relationship between the criterion and the predictor(s) is linear rather than curvilinear. Plot #1 is the residual plot for which errors (y-axis)) plotted as a function of predicted/fitted values (x-axis); a straight horizontal line on this plot would indicate that the linearity assumption is met; ff a curvilinear model is better than a linear model, then the errors would not be distributed normally, but instead be greater in some parts of the plot than in others.

6.1.5 Homoscedasticity of the residuals. The variance in the residuals, or error in prediction, should be the same across the values of the prectors. Homoscedasticity is also referred to as constant variance. Non-constant variance in the residuals is referred to as heteroscedasticity. We can examine homoscedasticity visually and statistically.

Plot #1 can also provide information about the variance in the residuals across levels of the predictor. Based on this example, you can see that there is more variability of the residuals (y-axis) on the left side of the plot than at the right side; thus the variability in residuals does not seem to be constant. If the variance is not constant, then we may have issues with residuals not being distributed normally (see next assumption).

If we want to statisically test for constant variance, we can use the ncvTest() function from the car library (install if you haven't yet). The ncv stands for non-constant variance. We want constant variance. The ncvTest() function provides a Chi-Square test value and a corresponding *p*-value for the test. If p is less

than or equal to alpha, you have evidence that that you have non-constant variance, or heteroscedasticity. Unfortunately, we have heteroscedasticity.

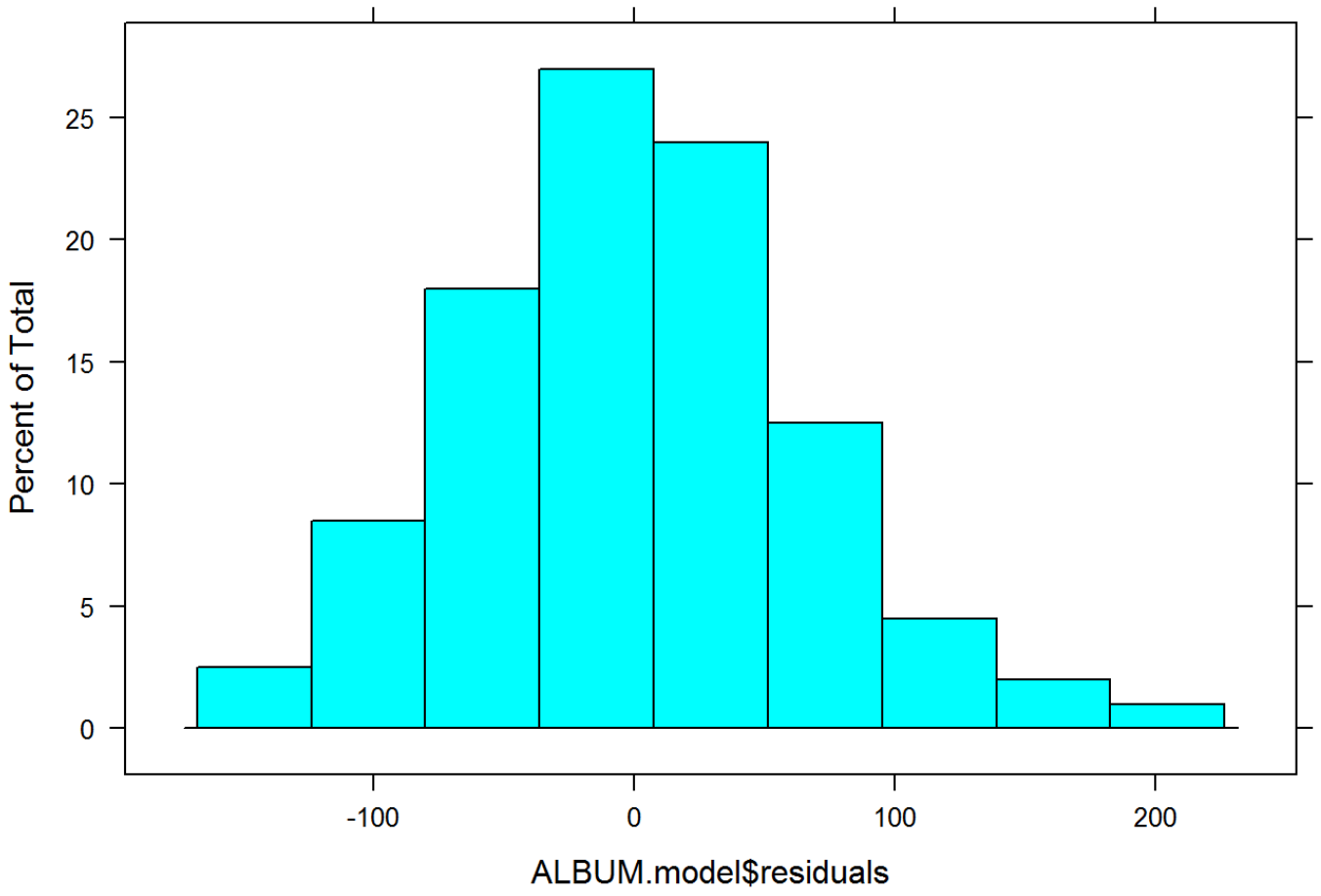
```
#library(car) # load if not already loaded
ncvTest(ALBUM.model) # the p-value is much lower than .05
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 8.233055    Df = 1     p = 0.004113423
```

6.1.6. Residuals (prediction errors) are distributed normally. If errors are random and the size of the prediction error does not depend on whether the score on the predictor is low or high, then those errors should be distributed normally. Plot #2 will provide detail about the normality of the residuals. Plot #2 is a quantile-quantile plot used to determine normality of errors; if points fall along the identity line, the errors are normally distributed. In this example, you can see that points on the ends of the distribution do not quite fall along the identify line. We could also test this assumption by using the sharipo.test() function on the residuals themselves.

We can take a look at a histogram or because the residuals are actually stored as part of the information returned from running the lm() function, we can extract those residuals from the model object. Lukily, they are named appropriately as residuals. Remember, the model was ALBUM.model; the residuals do not depart from a normal distribution.

```
histogram(~ ALBUM.model$residuals) # take a look; looks fairly normal to the eye
```




```
shapiro.test(ALBUM.model$residuals) # test for normality; p > alpha so the residuals do n
ot depart far enough from normality to say we have violated this assumption.
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ALBUM.model$residuals
## W = 0.98995, p-value = 0.1757
```

6.1.7. No multicolinearity; predictors should not be strongly correlated with each other. When you have more than one predictor, you have to examine for multicolinearity. collinearity will mask the true relationship among variables. Having multicolinearity makes the beta values (y-intercept and slope) unreliable and untrustworthy when making inferences from samples to populations. Generally speaking, there will be more error in beta values for greater amounts of multicolinearity. There are other influences too, but we won't address them here. You should just know to determine if you meet this assumption. *Because we only have one predictor in this example, we do not have to test for this assumption.*

If you had multiple predictors, you would want to test for multicollinearity. One easy way to test this is to examine Variance Inflation Factors (VIFs) using the vif() function from the car library. The function returns a VIF value for each predictor. If the square root of the VIF is greater than 2, this predictor would be eliminated from your model.

An example if you have multiple predictors:

```
# vif(mymodel) # variance inflation factors for the model
# sqrt(vif(mymodel)) # the square root of them
```

6.1.7. Independence of errors. For any two obervations in the data, the errors are not related. This is usually not a problem as long as your sample is selected at random and you don't allow people to participate in your study, allow them to talk to other participants, etc. in ways that affect how two people (or objects) respond. This independence assumption can be tested using the Durbin-Watson test using the durbinWatsonTest() function in the car library. The Durbin-Watson test will produce a D-W statistic value that will be large if independence is violated (and you have dependence). Compare p to alpha. In this example, we do not see violations of independence, which is good.

```
durbinWatsonTest(ALBUM.model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.04394305      2.032324   0.832
##  Alternative hypothesis: rho != 0
```

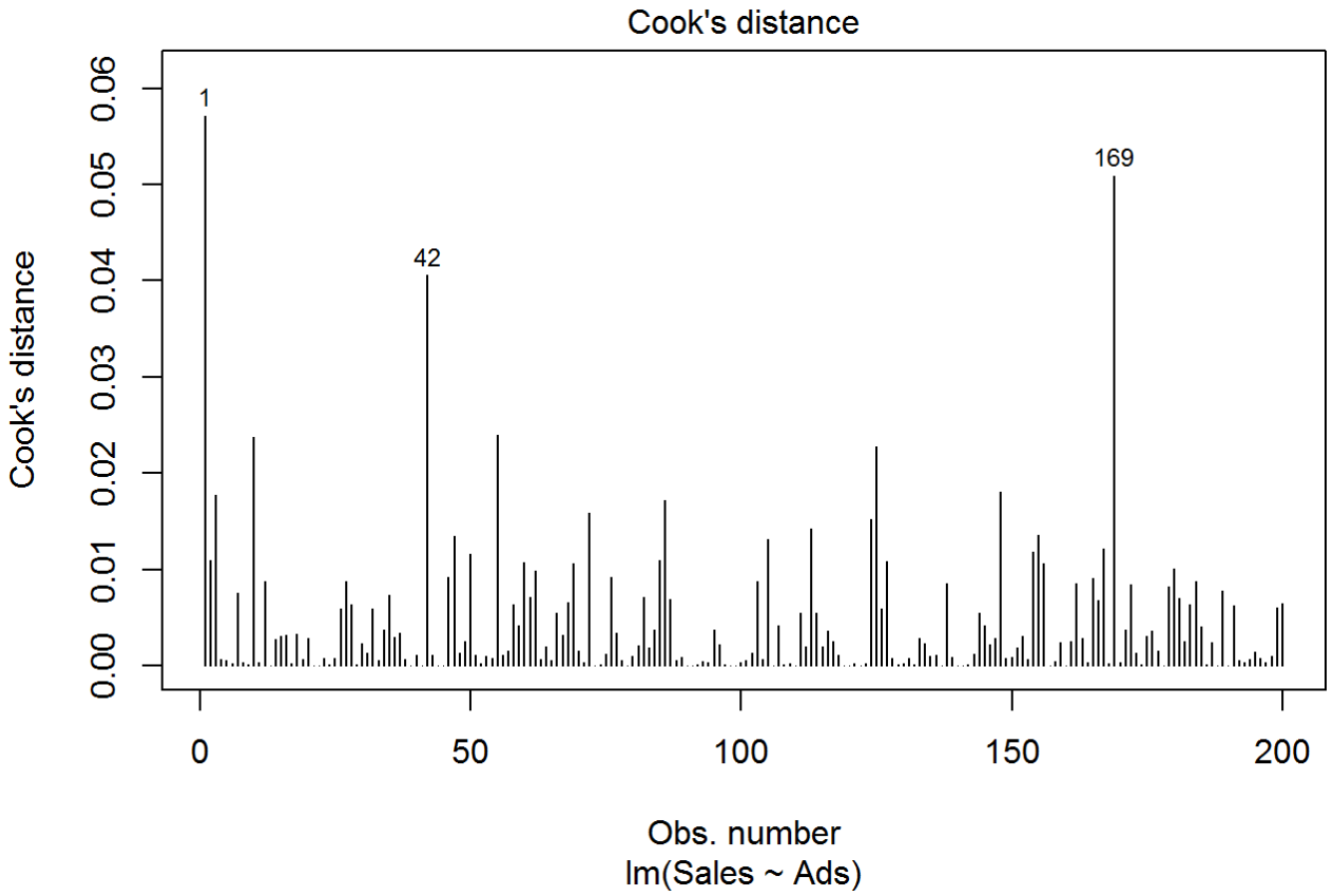
7. Determining Outliers

When building a linear model, it is always important to determine if there are any outliers in your data. Any interpretation of your regression model is useful to the extent that you don't have any outliers in your data set that inlfuence your correlation or the predictive power either by making it stronger than it actually is or weakening it.

7.1. Cook's Distance measure is easy to use and actually tests how each data point affect your regression model by providing a Cook's distance value. It combines the information of leverage (how a point changes your slope) and residual of the observation (the prediction error). If a data point influences your model, it will let you know. We can flag data points as outliers (which would would want to remove) if they have large Cook's D values.

7.2. We will be using Cook's distance to determine the influence that a data point has on your linear model. Cook and Weisberg (1982) suggest that Cook's D values > 1 are outliers. The code for doing so is a little complicated, but it will plot Cook's D values for the model as a function of rows in the data frame and it flag the cases/rows with the highest Cook's D values. Just because they are flagged, this does not mean you need to remove these rows from the data frame.

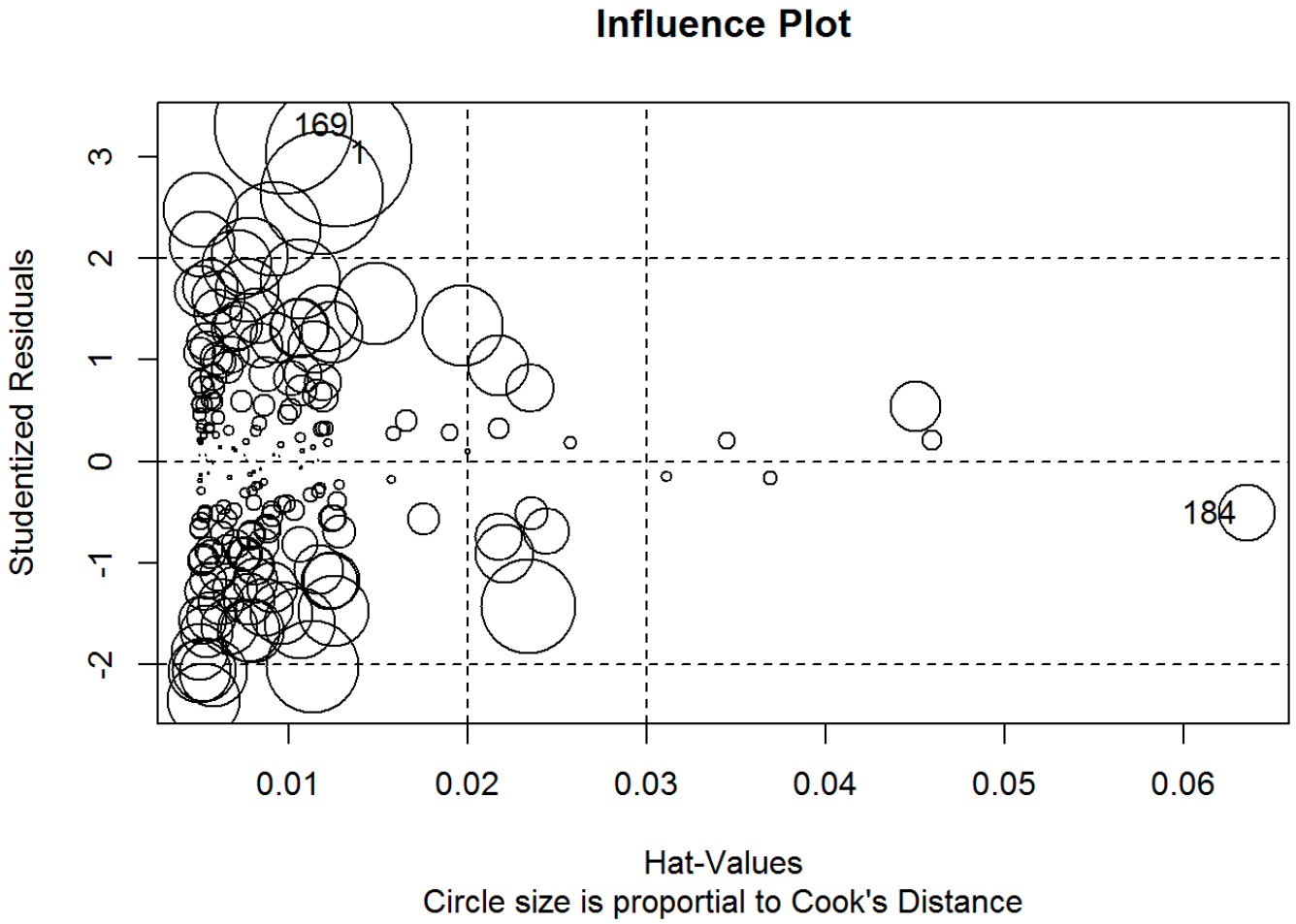
```
plot(ALBUM.model, which = 4, cook.levels = cutoff) # check to see if D values are greater
than 1
```



7.3. BONUS: The influencePlot() function from the car library can be useful to flag individual influence points. The plot displays *hat values* or *leverage* points. The size of the circle in the plot is related to the amount that the data point influences (or has leverage on your model). If you specify the id.method argument as “noteworthy” you can find out which row in your data frame is of concern because the row number will appear by a circle. You do not want to include data points that influence your model. You can remove them from your data set by creating a subset data frame to manipulate.

Leverage or hat values near 0 indicate little or no influence on your model, whereas those near 1 indicate complete influence. Belsley et al. (1980) suggest that values greater than 2 times the average are typically problematic.

```
influencePlot(ALBUM.model,
              id.method = "noteworthy",
              main = "Influence Plot",
              sub = "Circle size is proportial to Cook's Distance")
```




```
##       StudRes         Hat       CookD
## 1    3.032981 0.012776593 0.057159414
## 169  3.306071 0.009682918 0.050883173
## 184 -0.508298 0.063528920 0.008796577
```

In this graph the x-axis refers to the data point while the y-axis refers to cook's distance, telling us how influential each data point is. It may appear data points 6 and 7 are the most influential but the y-axis is very small with the highest data point being slightly above .6 cooks distance. We can determine that our original data set did not have any extremely influence single data points. If any data points are above a 1 on Cook's distance, it is up to you to remove them using previous methods of removing outliers.

8. Multiple Linear Regression

Multiple regression focuses on using more than one predictor to predict a criterion. Because predictors need to be correlated with the criterion to be useful, examining the correlations of the variables is useful.

```
round(cor(ALBUM[, -1]), 3)  # remove column 1 of ALBUM because ID is the first column in
the data frame
```

```
##          Sales   Ads Airplay Attract Ads.sqrt
## Sales    1.000 0.578   0.599   0.326    0.551
## Ads      0.578 1.000   0.102   0.081    0.971
## Airplay  0.599 0.102   1.000   0.182    0.068
## Attract  0.326 0.081   0.182   1.000    0.081
## Ads.sqrt 0.551 0.971   0.068   0.081    1.000
```

8.1. **QUESTION:** What is the correlation value between Sales and Airplay?

ANSWER: r = .59

8.2. **QUESTION:** Use the cor.test() function to test whether the correlation between Sales and Airplay is statistically different from 0; use alpha = .05. Provide the *p*-value in your answer.

ANSWER:

```
cor.test(ALBUM$Sales, ALBUM$Airplay)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ALBUM$Sales and ALBUM$Airplay
## t = 10.524, df = 198, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5018814 0.6810668
## sample estimates:
##       cor
## 0.5989188
```

Sales and Airplay r = .599; p < .001

8.3. **QUESTION:** Use the cor.test() function to examine the correlations between the 3 predictors. Are any pairs of predictors correlated significantly?; use alpha = .05?

ANSWER:

```
cor.test(ALBUM$Ads, ALBUM$Airplay)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ALBUM$Ads and ALBUM$Airplay
## t = 1.4411, df = 198, p-value = 0.1511
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.03738668  0.23726994
## sample estimates:
##       cor
## 0.1018828
```

```
cor.test(ALBUM$Ads, ALBUM$Attract)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ALBUM$Ads and ALBUM$Attract
## t = 1.14, df = 198, p-value = 0.2557
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.05864656  0.21706067
## sample estimates:
##        cor
## 0.08075151
```

```
cor.test(ALBUM$Airplay, ALBUM$Attract)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ALBUM$Airplay and ALBUM$Attract
## t = 2.6043, df = 198, p-value = 0.009904
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.04436788 0.31283089
## sample estimates:
##       cor
## 0.1819886
```

Ads and Airplay = r = .101; p = .15 Ads and Attract = r = .081; p = .26 Ads and Airplay = r = .182; p = .01

You might have notice that two predictors are correlated significantly. In order to determine if a predictor is problematic to your regression model, you would want to check for multicollinearity.

9. Define a multiple-regression model

9.1. There are different types of models to create, but the focus will be on simple additive models. Taking a *hierarchical regression* approach, create the model by entering predictors in the order of importance or theoretical contribution.

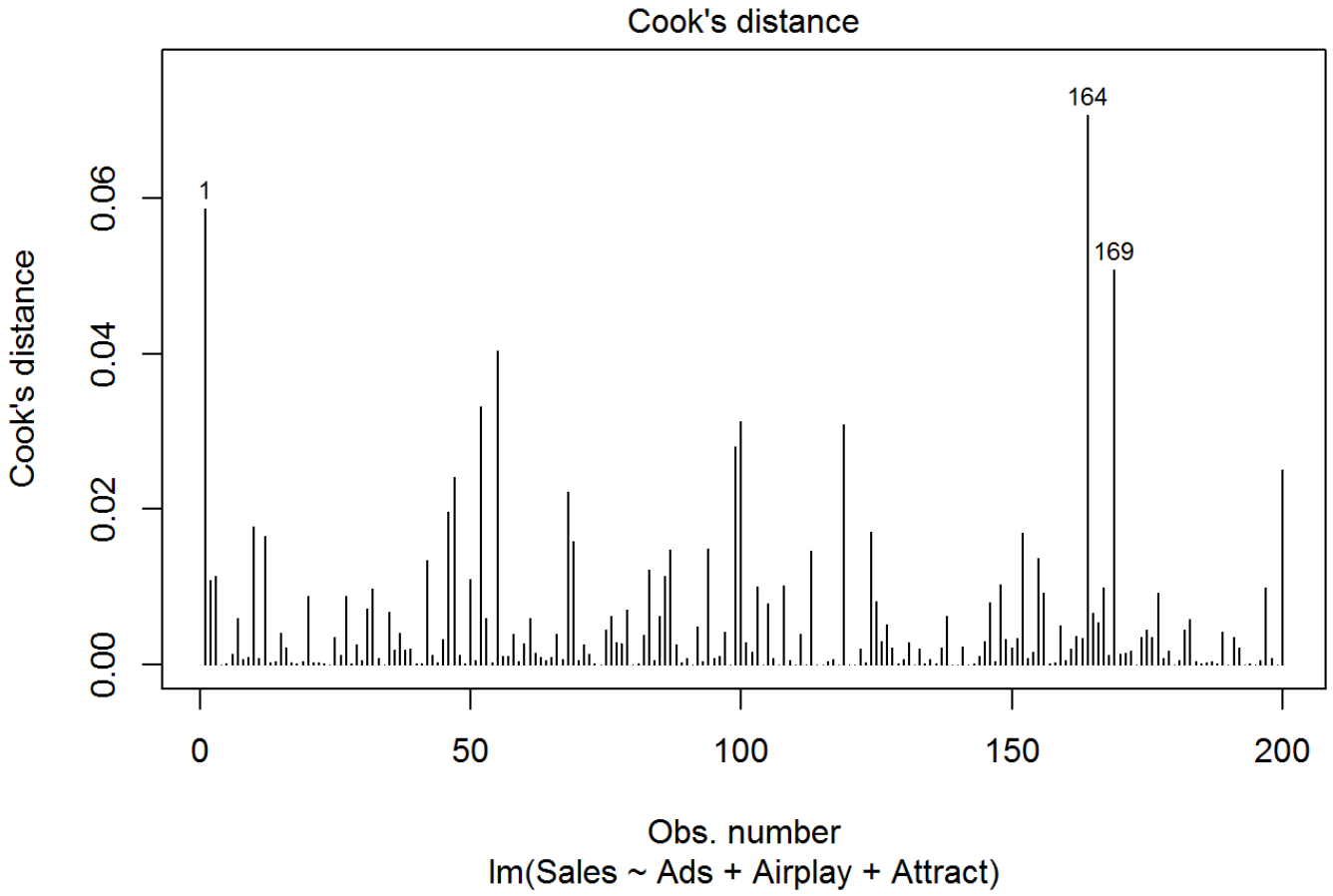
mymodel <- lm(formula = criterion ~ predictor1 + predictor3 + predictor3, data = mydataframe, na.action = some action)

9.2. Specify the multiple-regression model using multiple predictors. Use the lm() function to build a linear model to predict album sales from Ads, Airplay, and Attraction of band members and name the regression model ALBUM.model2 so that you can distinguish this from the bivariate model.

```
ALBUM.model2 <- lm(formula = Sales ~ Ads + Airplay + Attract,
              data = ALBUM,
              na.action = na.exclude)
```

9.3. **QUESTION:** Calculate Cook's D for the model to determine if there are any outliers that will influence model. Replace mymodel in the code below with the model object you just created.

```
plot(ALBUM.model2, which = 4, cook.levels = cutoff)
```



9.4. Obtain a summary of the model using the summary() function.

```
summary(ALBUM.model2)
```

```
##
## Call:
## lm(formula = Sales ~ Ads + Airplay + Attract, data = ALBUM, na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -121.324  -28.336   -0.451   28.967  144.132
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -26.612958  17.350001  -1.534    0.127
## Ads           0.084885   0.006923  12.261  < 2e-16 ***
## Airplay       3.367425   0.277771  12.123  < 2e-16 ***
## Attract      11.086335   2.437849   4.548 9.49e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.09 on 196 degrees of freedom
## Multiple R-squared:  0.6647, Adjusted R-squared:  0.6595
## F-statistic: 129.5 on 3 and 196 DF,  p-value: < 2.2e-16
```

QUESTION: Use the F-statisic and the *p*-value to help determine whether the model is statistically better than a mean-based model. Be careful with reading the scientific notation.

ANSWER: F(3, 196) = 129.5, p < .001. The multiple-regression model is a better predictor than the mean based model based on no predictor(s). The regression model seems to fit the data.

QUESTION: What proportion of the variance in Sales does the model account for? Is this greater than was accounted for by the simple model?

ANSWER: R-squared = .6647. This amount of variance accounted for my predictors is certainly better than not having predictors. Adjusted for bias in the sample, the poplulation adjusted R is slightly lower, but still .6595. Almost two-thrids of the variability in Sales is accounted for by the preditors together.

QUESTION: Are the regression coefficients (bs, slopes) for the 3 predictors significantly different from a slope of 0?

ANSWER: Yes. b1(ads) has a large t-value; p < .05; b2(Airplay) has a large t-value; p < .05; b3(Attract) does not appear to have as large of a t-value; but p < .05. All slopes are positive, which makes sense because all predictors are correlated positively with the criterion.

QUESTION: Which predictor has the steepest standardized slope value?

ANSWER: Cannot answer yet. I don't know how to standardize betas.

9.5. Standardized beta coefficients adjust the slope estimate as a function of the standard error of the estimate. This is one way to compare the regression coefficients of different predictors. The lm.beta() function from the QuantPsyc library will standarize the values in order to compare them easily. Here is an

example:

```
#library(QuantPsyc) # load if not already loaded
ALBUM.model2.betas <- lm.beta(ALBUM.model2)  # assign the standardized betas to an object
ALBUM.model2.betas # take a look at the object to compare the betas
```

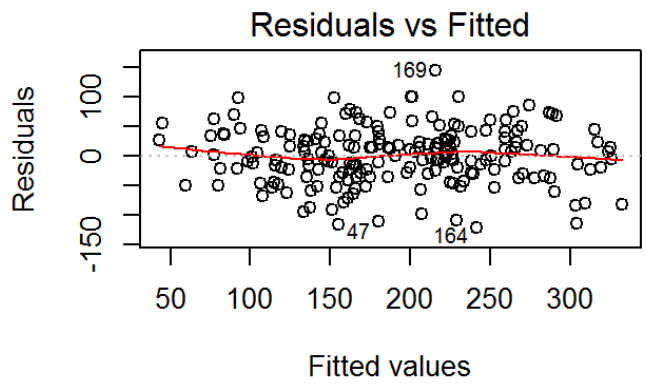
```
##       Ads   Airplay   Attract
## 0.5108462 0.5119881 0.1916834
```

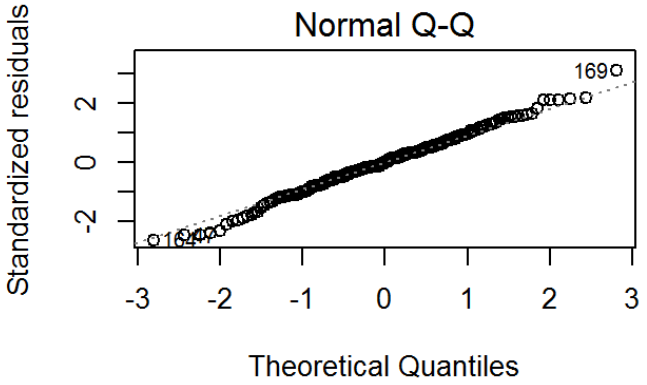
QUESTION: Now that the slopes have been standarized, you can examine the change in the criterion for each standard deviation change in each predictor. Which predictor has the steepest standardized slope value (biggest bang for your buck)?

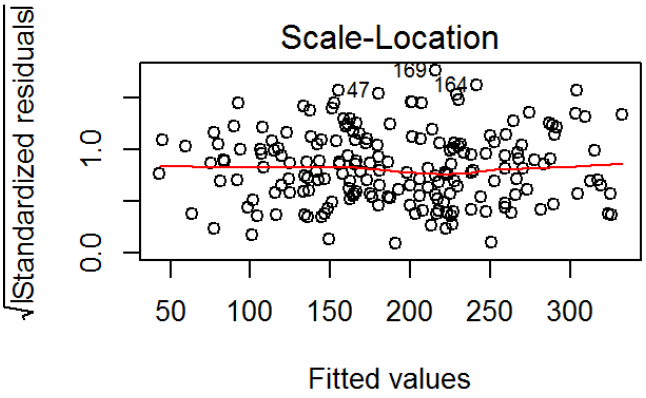
ANSWER: Ads and Airplay both have larger standarized changes in Sales (criterion) for each standard deviation change in them (predictors) relative to Attraction of the band members. Ads and Airplay both are very close in standardized betas (unique variance), but Airplay seems to provide the biggest influence on Sales. However, convincing radio DJs to play crappy songs might be more challenging than advertising. Only you can determine which approach is most effective to use, given your goals and your resources.

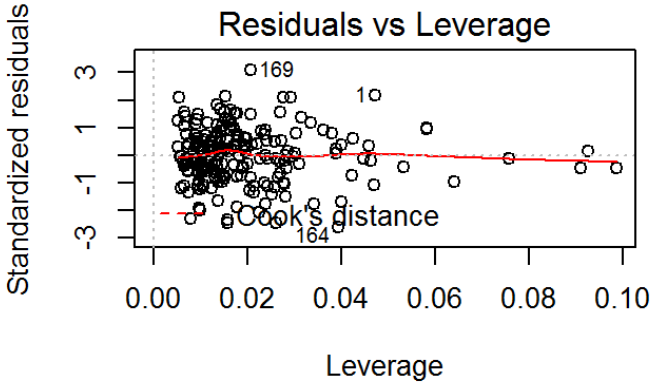
9.6. Use the plot() function to generate the diagnostic plots for examining the model. Step through the plots one at a time might be helpful to see them on a larger scale.

```
par(mfrow = c(2,2))
plot(ALBUM.model2)
```










```
par(mfrow = c(1,1))
```

QUESTION: Based on that plot, does the model appear to be a linear model? Explain how you arrived at that decision.

ANSWER: Plot #1 shows a relatively straight line for the residuals plotted as a function of predicted values. It's not a straight as it could be, but it's certainly not an obvious curve. Perhaps there are outliers to remove that will be good to remove.

9.7. **QUESTION:** Use the appropriate function to test for the homoscedasticity/constant variance assumption. Is the assumption met? Why or why not?

```
ncvTest(ALBUM.model2)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.3030143    Df = 1     p = 0.5819989
```

ANSWER: The test for non-constant variance has a small Chi-square value with a corresponding p-value of .58 ; p > .05. Thus, we don't have heteroscedasticity.

9.8. **QUESTION:** Use the appropriate function to test to determine whether the residuals of the model are distributed normally.

```
shapiro.test(ALBUM.model2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ALBUM.model2$residuals
## W = 0.99483, p-value = 0.7253
```

ANSWER: The W statistic is .994 with a corresponding *p*-value of .72; p > .05 so we don't have evidence that residuals deviate from a normal distribution

9.9. **QUESTION:** Use the appropriate function to test for independence of errors. Is the assumption met? Why or why not?

```
durbinWatsonTest(ALBUM.model2)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.0026951      1.949819   0.774
##  Alternative hypothesis: rho != 0
```

ANSWER: The D-W statistic value is about 1.95 with a corresponding p-value of .74; p > .05, so there does not appear to be a violation in independence of errors.

9.10. **QUESTION:** Use the appropriate function to test for multicolinearity. Use the suggested values to determine if you have multicolinearity.

```
vif(ALBUM.model2)
```

```
##      Ads  Airplay  Attract
## 1.014593 1.042504 1.038455
```

```
sqrt(vif(ALBUM.model2))
```

```
##      Ads  Airplay  Attract
## 1.007270 1.021031 1.019046
```

ANSWER: The square root of the variance inflation factors do not exceed 2. Although there is a correlation between some of the predictors, that relationship is not large enough to compromise estimates of unique variance for each predictor.

9.11. **BONUS QUESTION:** Compare the adjusted R-squared values for the bivariate model and the model with 3 predictors.

ANSWER:

Model 1: Adjusted R-squared: 0.3313 Model 2: Adjusted R-squared: 0.6595

We could also compare the models statistically using an ANOVA test. This shows that the model with 3 preditors is statisically better than the simple model.

```
anova(ALBUM.model, ALBUM.model2)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Ads
## Model 2: Sales ~ Ads + Airplay + Attract
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    198 862264
## 2    196 434575  2    427690 96.447 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```