Linear Regression Lab

This is the R Notebook version of the Linear Regression lab. This verison of the document allows for R code and text to coexist in the same script. The R script version of this file, which is how we operated the labs, is also available in canvas. You can learn more about R Notebook at this tutorial if you would like to utilize this functionality: https://rmarkdown.rstudio.com/r_notebooks.html

The first task when we open RStudio is to set our working directory. As covered in both of the optional labs, and in the documentation, navigate in the files window (bottom right) to your documents folder for this class. In "More", click "Set as Working Directory".

Now let's load in the dataset. You can either open it from your working directory (if you've saved it there) or pull it up with the below command. Change the file path to make sure it matches yours.

```
calschooldist <- read.csv("calschooldist.csv")
View(calschooldist)
```

We can rename our dataset as an object. You can name this whatever you want, but I recommend calschool because it will be in line with the code below.

```
calschool <- calschooldist
```

We need to load "packages" in R which include commands and functions that we can leverage. These packages will vary based on the lab. Today we are going to be using the below for the assignment. You only need to do this once:

Activate these packages: You need to do this every time you open R. We've only installed them, now we need to activate them for this instance.

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.4.3
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

If you're having trouble installing the car package, try the following code:

To see our numbers (particularly the p values) more clearly, we can remove scientific notation. We need to repeat this every time we open R.

```
options(scipen=999)
```

Now that our working directory is set, our packages are loaded, we can begin the steps outlined in the prompt.

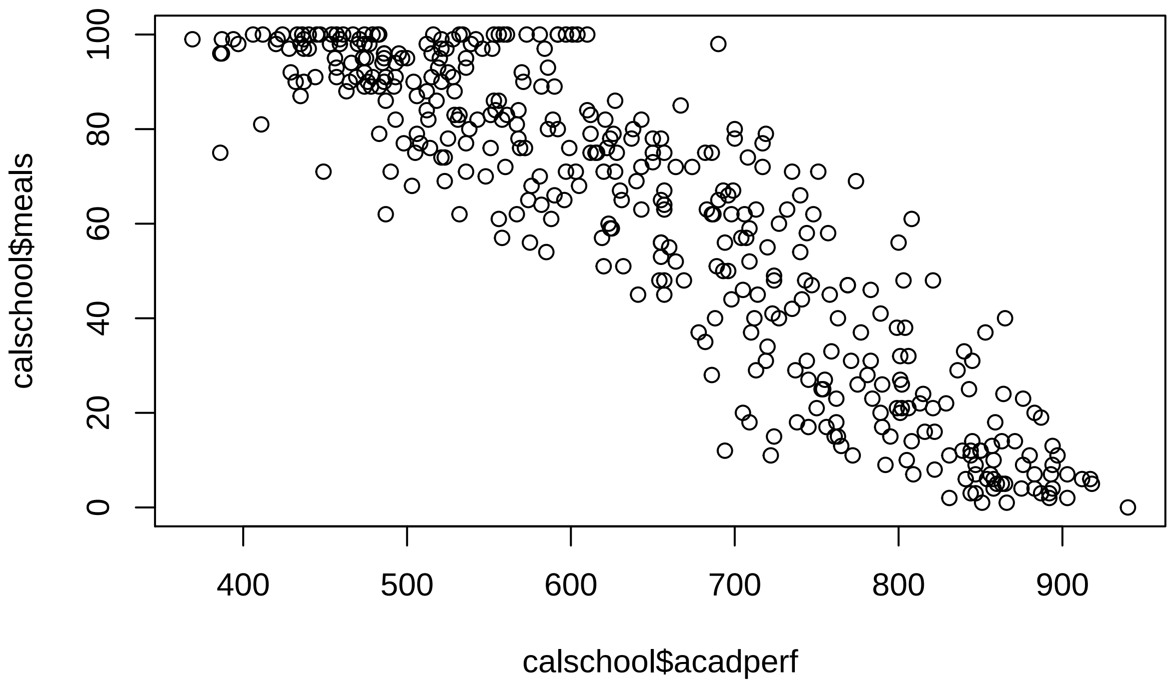
STEP 1: no R code to complete this

STEP 2: no R code to complete this

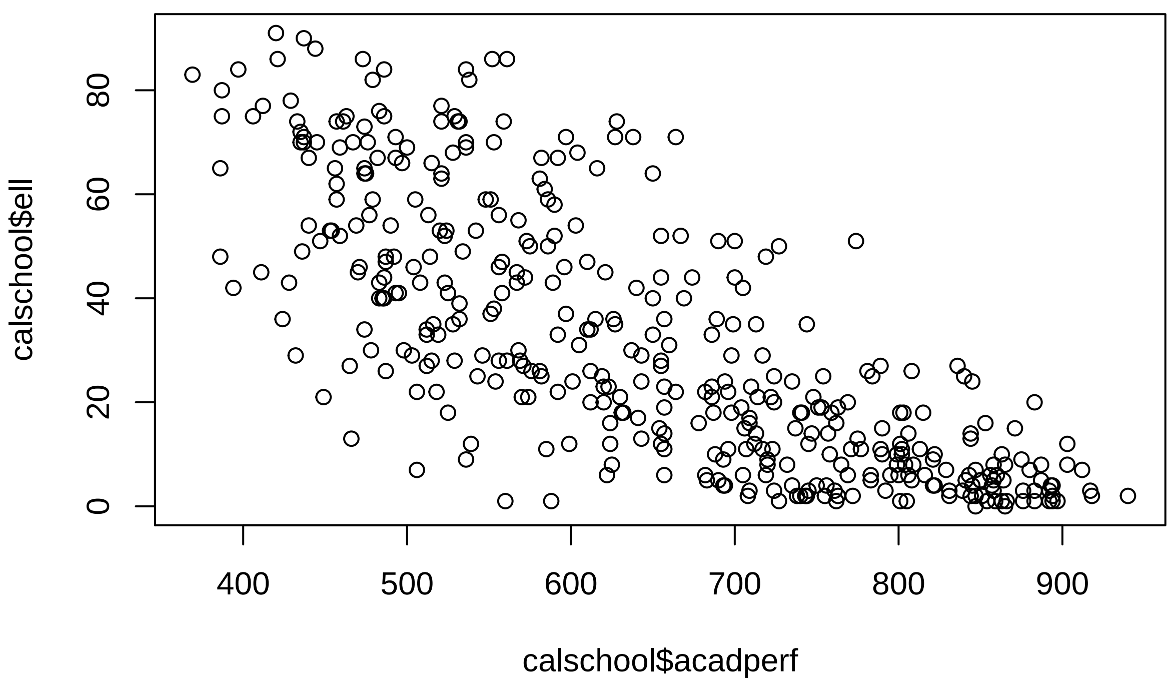
STEP 3: no R code to complete this

STEP 4: Perform basic checks of the candidate variables. Do you have any missing value or out of range data problems? If so, what did you do to resolve them, if anything? First, let's look at plots to get a feel for the data. In R, we can just use "plot" to create a scatterplot of two variables. We have to define those two items. Let's plot our dependent variable, acadperf, against some of the independent variables.

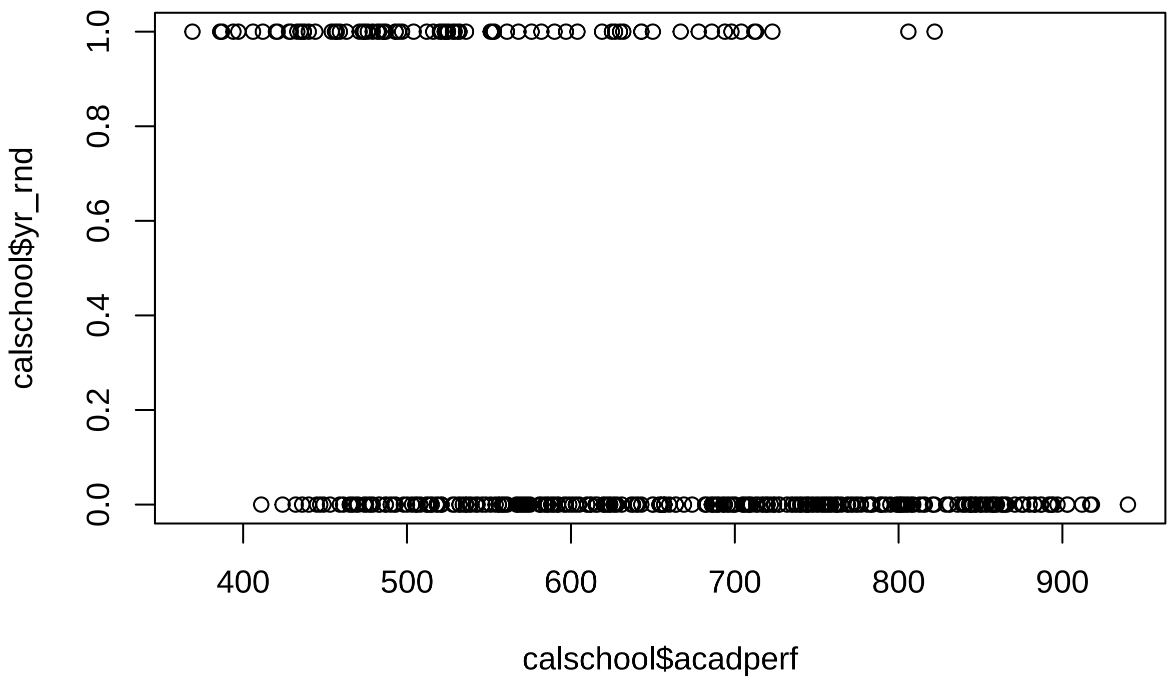
```r
plot(calschool$acadperf,calschool$meals)
```



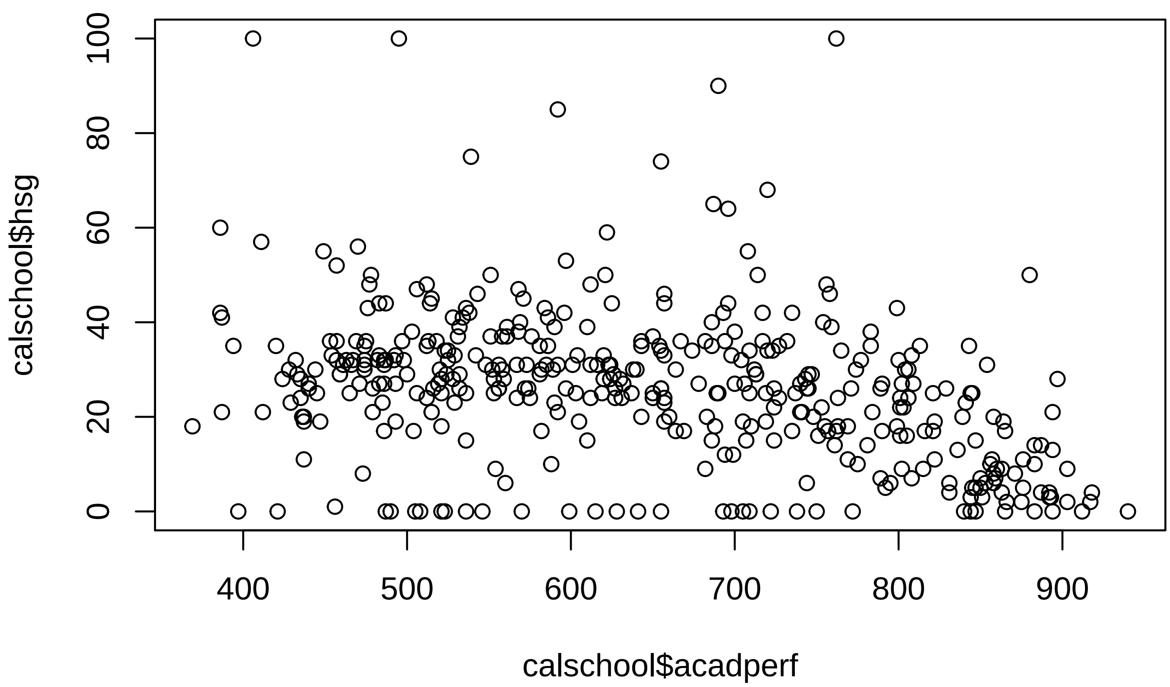

```r
plot(calschool$acadperf,calschool$ell)
```




```r
plot(calschool$acadperf,calschool$yr_rnd)
```




```r
plot(calschool$acadperf,calschool$hsg)
```



Have you noticed that any have a strong correlation? What does the graph of acadperf vs ell tell us about the ell data? Confirm this with the str code we completed in the lab. We can try a new function here, called "describe". This will give us new statistics that are a little different than "summary".

```r
describe(calschool)
```

```
##          vars   n    mean      sd median trimmed     mad min  max range
## snum        1 400 2866.81 1543.81 3007.5 2880.86 1894.02  58 6072  6014
## dnum        2 400  457.74  184.82  401.0  468.53  284.66  41  796   755
## acadperf    3 400  647.62  142.25  643.0  645.79  177.17 369  940   571
```

```
## meals        4 400   60.31   31.91   67.5   62.18   37.81   0  100   100
## ell          5 400   31.45   24.84   25.0   29.39   28.17   0   91    91
## yr_rnd       6 400    0.23    0.42    0.0    0.16    0.00   0    1     1
## mobility     7 399   18.25    7.48   17.0   17.66    5.93   2   47    45
## acs          8 398   19.16    1.37   19.0   19.21    1.48  14   25    11
## not_hsg      9 400   21.25   20.68   14.0   18.65   19.27   0  100   100
## hsg         10 400   26.02   16.33   26.0   25.29   13.34   0  100   100
## some_col    11 400   19.71   11.34   19.0   19.65   11.86   0   67    67
## col_grad    12 400   19.70   16.47   16.0   18.12   16.31   0  100   100
## grad_sch    13 400    8.64   12.13    4.0    5.85    5.93   0   67    67
## full        14 400   84.55   14.95   88.0   86.60   14.83  37  100    63
## emer        15 400   12.66   11.75   10.0   11.14   10.38   0   59    59
## enroll      16 400  483.46  226.45  435.0  459.41  202.37 130 1570  1440
## mealcat     17 400    2.02    0.82    2.0    2.02    1.48   1    3     2
##            skew kurtosis    se
## snum      -0.01    -1.03 77.19
## dnum      -0.35    -0.78  9.24
## acadperf   0.10    -1.13  7.11
## meals     -0.41    -1.20  1.60
## ell        0.57    -0.87  1.24
## yr_rnd     1.28    -0.37  0.02
## mobility   0.83     1.14  0.37
## acs       -0.23     1.64  0.07
## not_hsg    0.99     0.44  1.03
## hsg        0.95     3.08  0.82
## some_col   0.25     0.13  0.57
## col_grad   1.47     4.32  0.82
## grad_sch   2.16     4.72  0.61
## full      -0.97     0.17  0.75
## emer       1.06     0.76  0.59
## enroll     1.34     3.02 11.32
## mealcat   -0.03    -1.51  0.04
```

Here we can see missing values, ranges, measures of central tendency, and standard deviation. What are your inferences about these values? I can see that most variables include 400 total entries, but mobility and acs have less than 400. I have to remove the nulls with the following code:

```r
calschooldist2=na.omit(calschooldist)
describe(calschooldist2)
```

```
##            vars   n    mean      sd median trimmed     mad min  max range
## snum          1 398 2869.53 1539.25 3007.5 2881.53 1892.54  58 6072  6014
## dnum          2 398  457.71  184.90  401.0  468.07  284.66  41  796   755
## acadperf      3 398  648.47  142.08  643.0  646.85  176.43 369  940   571
## meals         4 398   60.16   31.91   67.0   61.96   38.55   0  100   100
## ell           5 398   31.29   24.80   25.0   29.22   28.17   0   91    91
## yr_rnd        6 398    0.23    0.42    0.0    0.17    0.00   0    1     1
## mobility      7 398   18.26    7.49   17.0   17.67    5.93   2   47    45
## acs           8 398   19.16    1.37   19.0   19.21    1.48  14   25    11
## not_hsg       9 398   21.19   20.70   14.0   18.59   19.27   0  100   100
## hsg          10 398   25.99   16.37   26.0   25.24   13.34   0  100   100
## some_col     11 398   19.71   11.36   19.0   19.64   11.86   0   67    67
## col_grad     12 398   19.74   16.50   16.0   18.18   16.31   0  100   100
## grad_sch     13 398    8.66   12.16    4.0    5.92    5.93   0   67    67
## full         14 398   84.63   14.86   88.0   86.63   14.83  37  100    63
```

```
## emer       15 398   12.62   11.67   10.0   11.14   10.38   0   59    59
## enroll     16 398  483.21  226.98  433.0  459.19  203.12 130 1570  1440
## mealcat    17 398    2.01    0.82    2.0    2.01    1.48   1    3     2
##            skew kurtosis    se
## snum      -0.01    -1.02 77.16
## dnum      -0.35    -0.78  9.27
## acadperf   0.09    -1.13  7.12
## meals     -0.41    -1.21  1.60
## ell        0.59    -0.84  1.24
## yr_rnd     1.27    -0.39  0.02
## mobility   0.83     1.13  0.38
## acs       -0.23     1.64  0.07
## not_hsg    0.99     0.45  1.04
## hsg        0.95     3.06  0.82
## some_col   0.25     0.12  0.57
## col_grad   1.47     4.29  0.83
## grad_sch   2.15     4.68  0.61
## full      -0.97     0.18  0.74
## emer       1.06     0.80  0.58
## enroll     1.34     3.00 11.38
## mealcat   -0.02    -1.51  0.04
```

Now `calschooldist2` does not include any null values. Lets replace calschool so we don't have these NA values.

```
calschool <- calschooldist2
```

Step 5: What did your check of the correlation matrix find? Did you add any variables to the end of you list based on it? Does it look like you need to worry about multicollinearity? Do you remember this from the lab? This is a function that gives us not only our Pearson's Coefficients, but also gives us our p-values too! This function will be loaded into R, then when we run cor.prob with our data, we'll get the output. When looking at output generated by cor.prob, remember, p values are above and coefficients are below the diagonal line.

```
cor.prob <- function (X, dfr = nrow(X) - 2) {
  R <- cor(X, use="pairwise.complete.obs")
  above <- row(R) < col(R)
  r2 <- R[above]^2
  Fstat <- r2 * dfr/(1 - r2)
  R[above] <- 1 - pf(Fstat, 1, dfr)
  R[row(R) == col(R)] <- NA
  R
}
```

Now that we have added the function, let's run it

```
correlation_table_calschool <- cor.prob(calschool)
View(correlation_table_calschool)
```

You can save this table to your working directory with the following code:

```
write.csv(correlation_table_calschool, file = "Correlation Matrix California Schools.csv")
```

Are there any noteworthy correlations that might help you build your model? Which variables have the strongest relationships with academic performance? Would these variables be good to include in a regression analysis?

STEP 6: no R code to complete this

STEP 7: REGRESSION! Add your first independent variable. Show your bivariate / unadjusted model. Did

it accord with your expectations? Now we'll see a new command, "lm". This will fit a simple regression model to the data. The format for this code is lm(y~x, data) where y is the response (dependent variable), x is the predictor (independent variable), and the data is calschool.
Give it a try with your first variable, but YOU MUST CHANGE the variable name where I have [your first variable]:

This is what it would look like if "meals" was your first variable:

```
regression_1 <- lm(acadperf ~ meals, data = calschool)
```

Let's review the summary of this regression:

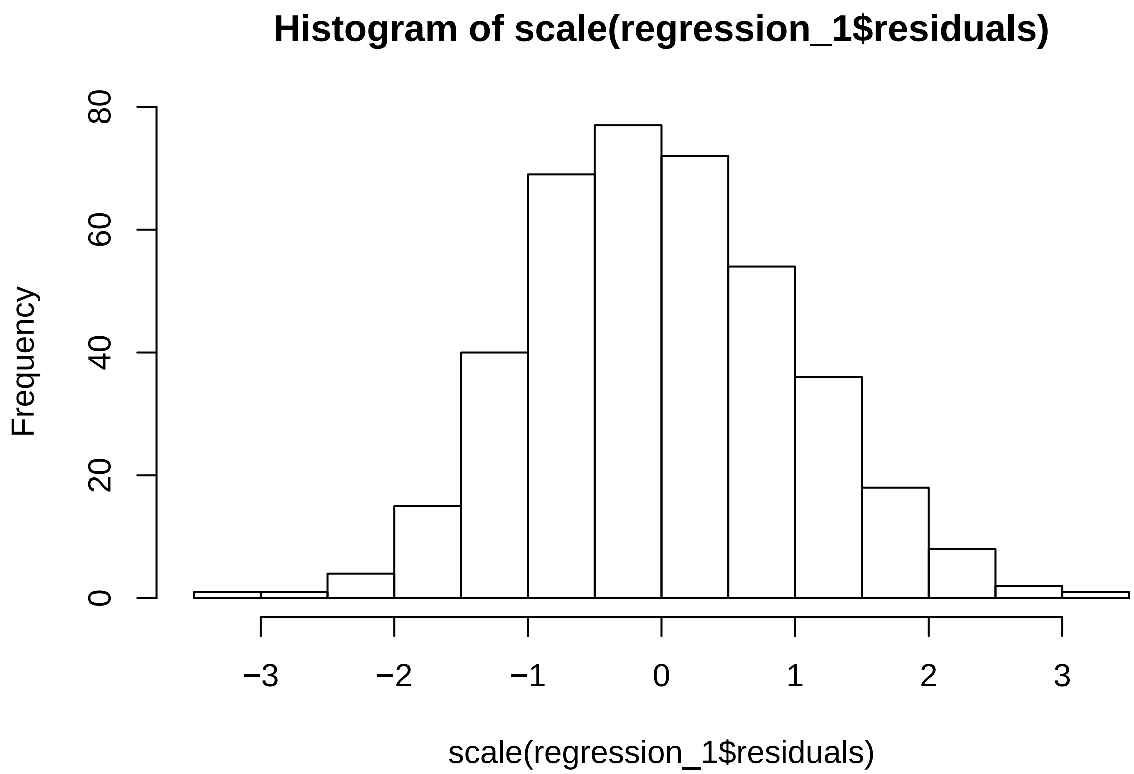
```
summary(regression_1)
```

```
##
## Call:
## lm(formula = acadperf ~ meals, data = calschool)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -202.99  -41.22   -3.45   44.30  193.19
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 889.60032    6.63315  134.11 <0.0000000000000002 ***
## meals        -4.00810    0.09743  -41.14 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.95 on 396 degrees of freedom
## Multiple R-squared:  0.8104, Adjusted R-squared:  0.8099
## F-statistic:  1692 on 1 and 396 DF,  p-value: < 0.00000000000000022
```

This tells us the results of our regression. Is it in line with your expectations?

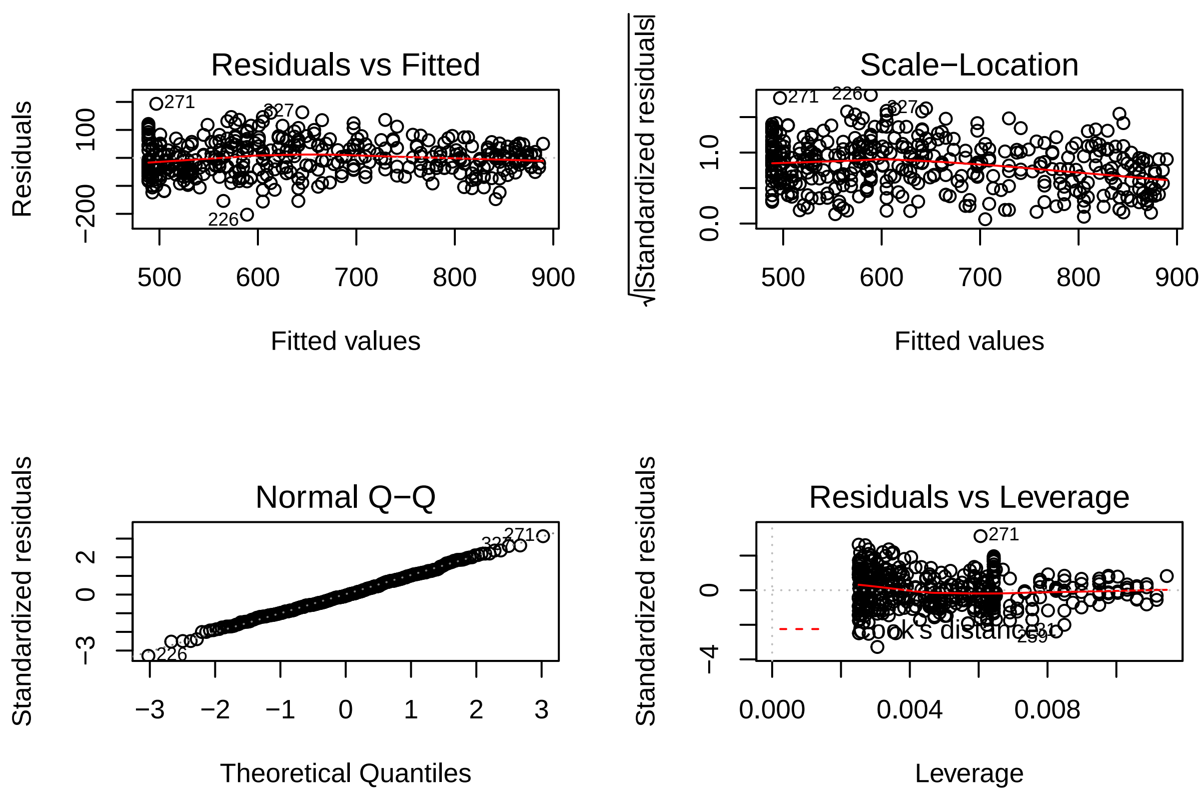
STEP 8: Check for regression violations for this bivariate mode. Did you find any major violations? To analyze residuals and test assumptions, we can explore some graphs. We'll start with a histogram of the residuals.

```
hist(scale(regression_1$residuals))
```



Let's tell R to layout our graphs in a matrix so we can easily view 4 graphs at once. The plot() function graphs 4 helpful plots for us.

```
layout(matrix(c(1,2,3,4),2,2))
plot(regression_1)
```



What if we want standardized coefficients? R is a little difficult in that it doesn't give them as part of the standard output. By using "scale" in our lm function, we can standardize the unit of analysis to compare coefficients. Input your first variable.

Mine will look like this:

```
lm(scale(acadperf) ~ scale(meals), data = calschool)
```

```
##
## Call:
## lm(formula = scale(acadperf) ~ scale(meals), data = calschool)
##
## Coefficients:
##            (Intercept)            scale(meals)
## -0.0000000000000003511  -0.9002098779509647430
```

How can the standardized coefficient be interpreted? How about standardized residuals? We'll start by viewing all residuals. R has a function "names()" so we can learn more about what information is available.

```
names(regression_1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

You can see that we have access to the coefficients and residuals of this regression. This can be an easy way to look only at the relevant data.

```
coefficients(regression_1)
```

```
## (Intercept)       meals
##  889.600316   -4.008099
```

We can also specify the entire set of residuals of each point (residual = predicted value - actual value)

```
regression_1$residuals
```

```
##            1            2            3            4            5
##   71.9423263   49.1448047   45.1853004   42.1286064  -54.8794927
##            6            7            8            9           10
##    8.4806754   48.4401797  -50.5841177   -9.5598203  -36.3654410
##           11           12           13           14           15
##  -34.5922168  -69.0252772  -30.5760186  -72.4950272  -52.5436220
##           16           17           18           19           20
##   11.5778651   -5.5436220  -17.2763505  -44.5193246  -63.3006479
##           21           22           23           24           25
##  -42.1386651  -60.3654410   42.8613349  -90.0414755   25.9261280
##           26           27           28           29           30
##  -70.0009798  -72.0009798 -108.4545315  -83.9928806  -55.4545315
##           31           32           33           34           35
## -123.5112255   -2.0252772   35.9423263   -0.2277556  -35.3978375
##           36           37           38           39           40
## -115.0252772  -15.2439539   53.6021625  134.8532357   19.5049728
##           41           42           43           44           45
##   75.9099297   66.7074513   70.5940633   50.3996840  -61.5274237
##           46           47           48           49           50
##    9.4239814  -10.3897384   -3.4707298    0.5697659   -9.4707298
##           51           52           53           54           55
##    8.0962099   97.0071194  -89.9604841   -8.4383332  -36.3978375
```

```
##          56           57           58           59           60 
##  41.4563780  -34.7904022  -48.7904022    2.1529039  -40.9199884 
##          61           62           63           64           65 
##   4.1448047  -42.8713936  -91.8713936  -41.7904022  -26.9037901 
##          66           67           68           69           70 
##  -9.7904022  -64.7904022   19.1529039   39.0071194   21.9747228 
##          71           72           73           74           75 
## -33.0738720   62.9828220  -10.1467643   27.0071194  -30.9523849 
##          76           77           78           79           80 
##  77.8856323  -35.5355229   69.0476151  -28.0333763  -13.4140358 
##          81           82           83           84           86 
##  37.5130719  -74.0981694  -65.1872599  -68.4788289   46.4482788 
##          87           88           89           90           91 
##  21.4563780    2.4968737   34.4887745   20.4320806  -15.5679194 
##          92           93           94           95           96 
## -14.5436220  115.2095978   64.2095978   23.9180289  -40.8794927 
##          97           98           99          100          101 
## 131.0476151   57.1205073   35.6993521  118.0881108   67.7803435 
##         102          103          104          105          106 
##  73.0314168   72.2095978    4.0719125  108.2095978   15.0800116 
##         107          108          109          110          111 
##  60.0314168  -57.1062686  123.7884426   70.2095978    3.8046409 
##         112          113          114          115          116 
##  49.2014987   82.6669556   60.0638133  -37.8632944   44.9018306 
##         117          118          119          120          121 
## -73.8875918  -52.8713936   15.5697659  -76.7904022  -98.7985013 
##         122          123          124          125          126 
##  21.5940633   28.8532357  -55.7904022  -41.0171780  -12.9847815 
##         127          128          129          130          131 
## -12.0495746  -26.3816393  -33.8308979   11.1691021  -82.7904022 
##         132          133          134          135          136 
## -21.7985013  -17.1791608  -67.8632944   83.1853004  129.9747228 
##         137          138          139          140          141 
## -44.9766824    5.6507573   -5.7904022 -117.8227987   61.0071194 
##         142          143          144          145          146 
## -21.7904022  -61.0333763  -65.4626306  -60.8146996   92.2095978 
##         147          148          149          150          151 
##  10.1772013   51.4482788   19.9099297   -2.2682513  -66.9604841 
##         152          153          154          155          156 
## -15.9847815  -46.8389970   -9.1548634   27.1691021  -18.8227987 
##         157          158          159          160          161 
## -43.8066005   45.9099297  -33.7985013  -52.8308979   55.0719125 
##         162          163          164          165          166 
##-105.7985013  -76.8066005   -9.8227987   51.1853004  -21.3492427 
##         167          168          169          170          171 
## -19.8066005  -58.8794927   70.9342271  103.2095978   52.9909211 
##         172          173          174          175          176 
##-123.7985013  -55.7985013  -33.9847815  -91.8551953  -61.8066005 
##         177          178          179          180          181 
##  45.7803435  -24.8875918  -22.8066005    8.4968737   -5.9280875 
##         182          183          184          185          186 
##  -6.7904022   45.2095978  -34.8956910   11.0962099  -31.7904022 
##         187          188          189          190          191 
## -11.8308979  146.0395159   49.1205073  -18.8227987   28.2014987 
```

```
##          192          193          194          195          196 
## -118.8227987  -99.8066005    1.9261280  -72.8146996   56.5049728 
##          197          198          199          200          201 
##  -71.7985013   23.1853004   43.2095978   72.9828220   28.0638133 
##          202          203          204          205          206 
##  -63.8146996  115.9828220   17.0476151  -26.8066005  -14.7904022 
##          207          208          209          210          211 
##   77.9423263   68.0071194    1.6669556  -67.9361867  111.6993521 
##          212          213          214          215          216 
##  -19.5922168  -52.2358548    8.4968737   31.4563780  -76.4626306 
##          217          218          219          220          221 
##  -46.3492427   77.7722444   26.0071194  -45.8632944  -66.4788289 
##          222          223          224          225          226 
##  -28.2115574   14.4239814  -35.0657729  -70.9847815 -202.9928806 
##          227          228          229          230          231 
## -153.9442858  160.9585245   -1.5112255  -82.9766824  -29.1224668 
##          232          233          234          235          236 
##  -96.8713936 -105.8956910 -156.0252772 -105.4788289   40.4725763 
##          237          238          239          240          241 
##   14.0152185  -72.9766824   51.4887745  -59.8470961  -51.0819711 
##          242          243          244          245          246 
##   21.4158823   -6.5598203  -33.5760186   -7.5517212   -3.4302341 
##          247          248          249          250          251 
##    1.4320806   50.8451366  -49.8794927    6.8046409   60.9261280 
##          252          253          254          255          256 
##  136.0233176   45.9018306  -26.1143677  -41.2763505    5.9099297 
##          257          258          259          260          261 
##  -55.0738720  -53.1872599 -147.5031263 -109.0981694  -40.2115574 
##          262          263          264          265          266 
## -104.4383332   48.7641452  -89.1062686  -37.8066005   84.2095978 
##          267          268          269          270          271 
##   45.8613349  103.2095978   30.7965418   27.5778651  193.1933995 
##          272          273          274          275          276 
##  121.2095978   35.2014987  -88.1629625   41.5454685   94.9099297 
##          277          278          279          280          281 
##   -6.4302341   11.5130719  -11.5517212    4.6102616   -2.5031263 
##          282          283          284          285          286 
##    3.8127401  -24.5517212  -19.8389970   82.0638133   56.9018306 
##          287          288          289          290          291 
##  -22.8308979   23.0071194    8.9423263  -51.9685832  -27.8389970 
##          292          293          294          295          296 
##  -46.8551953   20.1853004   39.6588564   55.8775332   62.6345590 
##          297          298          299          300          301 
##   67.7803435   64.9018306   78.0314168   79.6507573   63.7317487 
##          303          304          305          306          307 
##  114.9990202  145.9747228  -27.7904022   99.8694340  135.7236495 
##          308          309          310          311          312 
##    2.0557142  106.9018306   39.0395159   -8.8308979   66.8370375 
##          313          314          315          316          317 
##  114.9342271    9.4320806   61.7074513   73.5535676  -31.8632944 
##          318          319          320          321          322 
##   69.1529039   53.0395159    3.1367056   27.7560461   14.9747228 
##          323          324          325          326          327 
##  105.7884426  -10.1467643   -8.0252772  -20.4383332  162.8937314 
```

```
##         328          329          330          331          332
##  -24.9280875 -114.0495746  -22.1710617    3.6264599  -25.4869281
##         333          334          335          336          337
##   41.9828220  123.0314168   -2.9361867  -34.4788289  -67.3168462
##         338          339          340          341          342
##   93.0071194   10.4158823   22.4725763  -31.4626306 -103.1386651
##         343          344          345          346          347
##  -90.1467643  -91.3735401  -31.3006479 -154.0981694  -35.4140358
##         348          349          350          351          352
##   19.6183607   26.9585245  112.2095978   47.0314168   29.5130719
##         353          354          355          356          357
##   27.8208392   73.5616668   33.7236495   -2.2763505   -8.9685832
##         358          359          360          361          362
##   45.7884426   26.7884426  -14.5112255   78.5859642   67.2095978
##         363          364          365          366          367
##   -6.5274237   37.0152185  -29.9361867   -7.8713936  -35.8308979
##         368          369          370          371          372
##   -7.8875918   44.6588564   86.8694340  -80.8632944    4.7641452
##         373          374          375          376          377
##   -4.5598203  -68.2358548  -73.5112255  -55.4302341  -79.4545315
##         378          379          380          381          382
##  -27.9280875    1.0800116  -24.8713936   23.0476151   82.0962099
##         383          384          385          386          387
##  -28.1224668  -52.7904022  -57.9037901   57.0800116   27.2095978
##         388          389          390          391          392
##  -63.8713936  -43.2115574   13.7398478  -41.0495746  -55.8632944
##         393          394          395          396          397
##   -9.8632944  -17.9361867   17.6507573  -36.3816393  -33.3249453
##         398          399          400
##   16.6102616   42.1933995   15.1933995
```

Next, we'll create a component of our regression to store our standardized residuals. We'll use the standardized residuals to determine if any have an absolute value greater than 2.

```r
regression_1$standardized.residuals <- rstandard(regression_1)
regression_1$large_residual <- regression_1$standardized.residuals >2 | regression_1$standardized.resid
sum(regression_1$large_residual)
```

```
## [1] 18
```

What is your interpretation of the results? Let's calculate the Durban-Watson statistic.

```r
dwt(regression_1)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2756506      1.445141       0
##  Alternative hypothesis: rho != 0
```

STEP 9: Sequentially build up the model adding variables in the order you specified (don't check reg. assumptions at each stage) To build variables into your model, continue to use the lm() function, and add the variable on the back side of the ~. You should update the name of the model so you can save each iteration in R. The form is below. Don't forget to replace with your variables in the order that you defined.

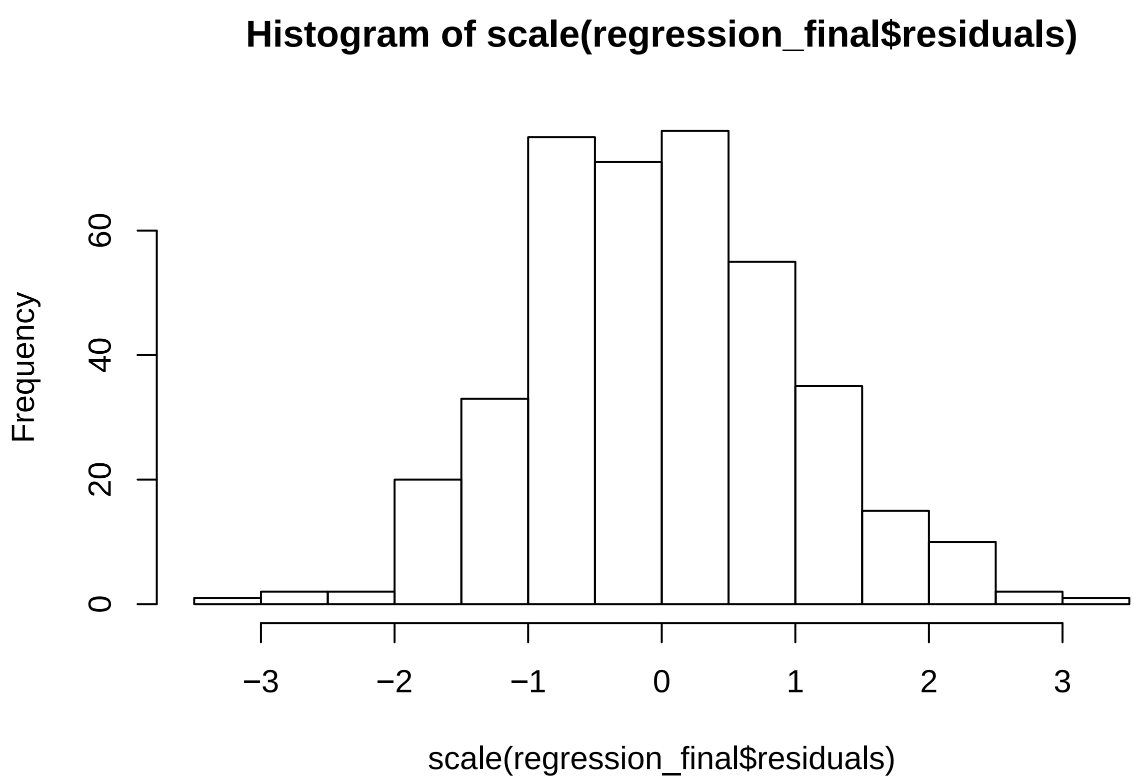
As an example, this is the form:

```r
regression_final <- lm(acadperf ~ meals + hsg + some_col, data = calschool)
summary(regression_final)
```

```
##
## Call:
## lm(formula = acadperf ~ meals + hsg + some_col, data = calschool)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -204.996  -42.817   -1.903   41.940  183.493
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 868.789886   9.595132  90.545 < 0.0000000000000002 ***
## meals        -3.939685   0.109528 -35.970 < 0.0000000000000002 ***
## hsg          -0.007379   0.208689  -0.035              0.97181
## some_col      0.856806   0.281680   3.042              0.00251 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.37 on 394 degrees of freedom
## Multiple R-squared:  0.8148, Adjusted R-squared:  0.8134
## F-statistic: 577.9 on 3 and 394 DF,  p-value: < 0.00000000000000022
```

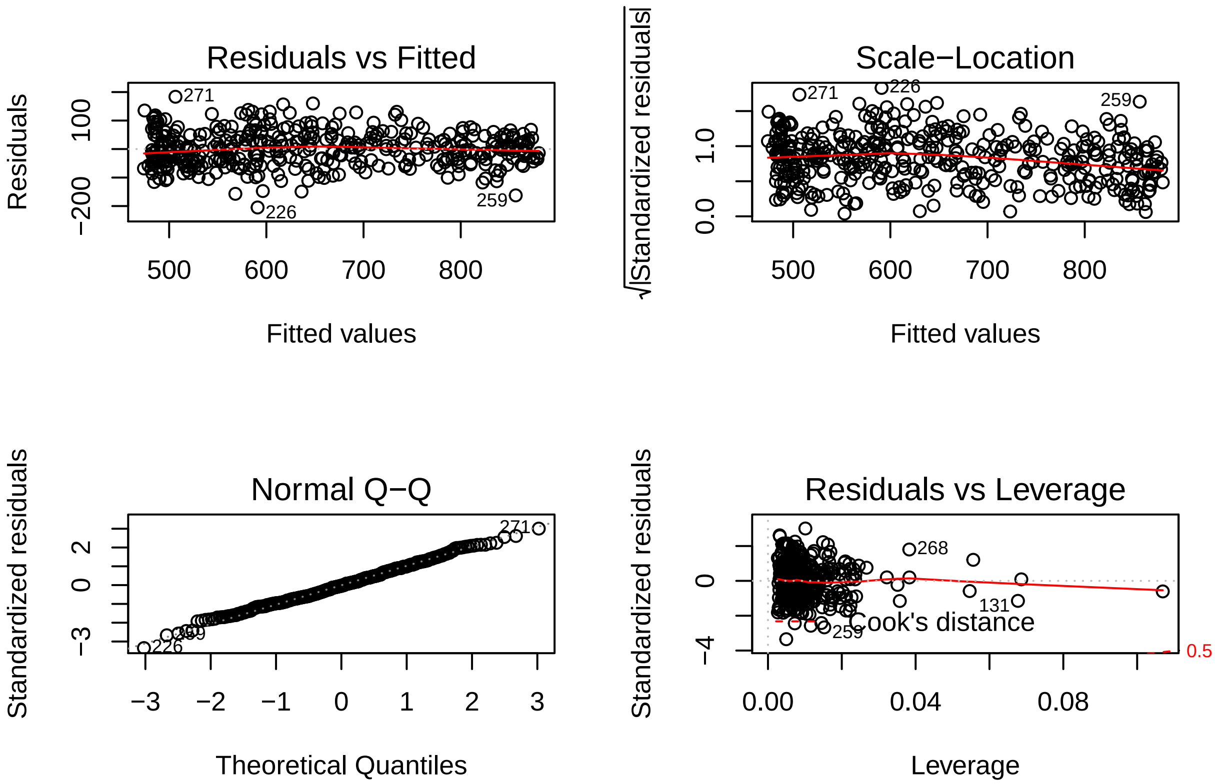
STEP 10: Recheck model assumptions Once we have our final model, let’s check assumptions through residual analysis as we did earlier.

```
hist(scale(regression_final$residuals))
```



View all plots at once

```
layout(matrix(c(1,2,3,4),2,2))
plot(regression_final)
```



Standardized residual analysis (how many residuals are more than two deviations away)

```
regression_final$standardized.residuals <- rstandard(regression_final)
regression_final$large_residual <- regression_final$standardized.residuals >2 | regression_final$standa
sum(regression_final$large_residual)
```

```
## [1] 18
```

Calculate the standardized coefficients as we did prior.

Mine will look like this:

```
lm(scale(acadperf) ~ scale(meals) + scale(hsg) + scale(some_col), data = calschool)
```

```
##
## Call:
## lm(formula = scale(acadperf) ~ scale(meals) + scale(hsg) + scale(some_col),
##     data = calschool)
##
## Coefficients:
##           (Intercept)            scale(meals)              scale(hsg)
## -0.0000000000000003583  -0.8848443174696255520  -0.0008499628631633799
##       scale(some_col)
##  0.0684972185835148323
```

You may have more (or fewer) variables in your model in comparison to my examples and that is okay. Just make sure all your variables in your final model are in standardized coefficients. Now that we have multiple terms in the model, lets include the multicollinearity test as well as Durbin-Watson.

```
vif(regression_final)
```

```
##    meals      hsg some_col
```

```
## 1.287600 1.229519 1.078979
```

```
dwt(regression_final)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2624994      1.469487       0
##  Alternative hypothesis: rho != 0
```

Discuss these results in your report and/or technical appendix. Don’t forget to include an advanced extension, in a different file.