MA678 Homework 2

9/26/2024

11.5

Residuals and predictions: The folder `Pyth` contains outcome y and predictors x_1, x_2 for 40 data points, with a further 20 points with the predictors but no observed outcome. Save the file to your working directory, then read it into R using `read.table()`.

```
pythData <- "https://raw.githubusercontent.com/avehtari/ROS-Examples/refs/heads/master/"
Pyth <- read.table(paste0(pythData, "Pyth/pyth.txt"), header = TRUE)
#pulling out data from dataset
```

(a)

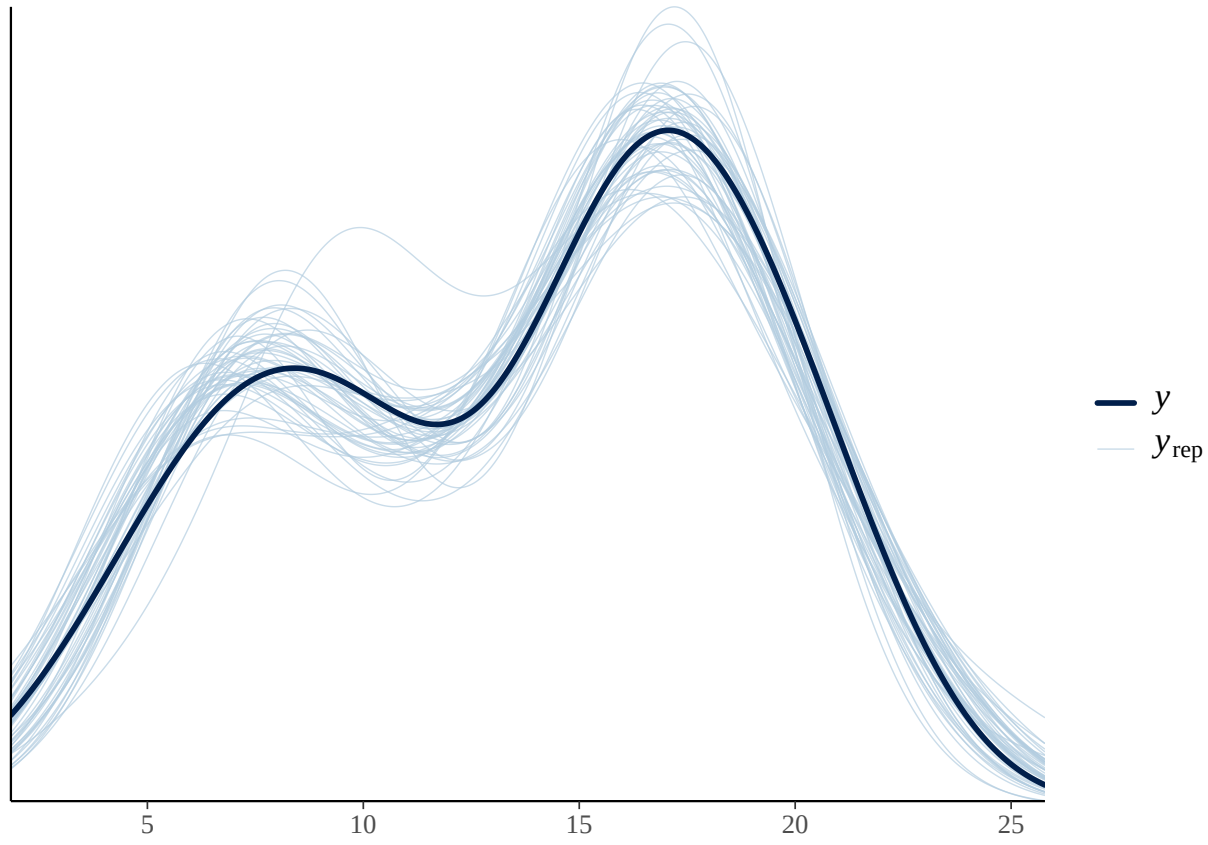
Use R to fit a linear regression model predicting y from x_1, x_2, using the first 40 data points in the file. Summarize the inferences and check the fit of your model.

```
LM = lm(y ~ x1 + x2, data = Pyth, subset = 1:40)
stanGlm = stan_glm(y ~ x1 + x2, data = Pyth, subset = 1:40, refresh = 0)
summary(stanGlm) #summarize the inferences
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x1 + x2
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 40
##  predictors:   3
##  subset:       1:40
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) 1.3    0.4  0.8   1.3   1.8
## x1          0.5    0.0  0.5   0.5   0.6
## x2          0.8    0.0  0.8   0.8   0.8
## sigma       0.9    0.1  0.8   0.9   1.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 13.6    0.2 13.3  13.6  13.9
##
```

```
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  4634
## x1            0.0  1.0  4254
## x2            0.0  1.0  5223
## sigma         0.0  1.0  3346
## mean_PPD      0.0  1.0  4047
## log-posterior 0.0  1.0  1587
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
pp_check(stanGlm) #check the fit of your model
```

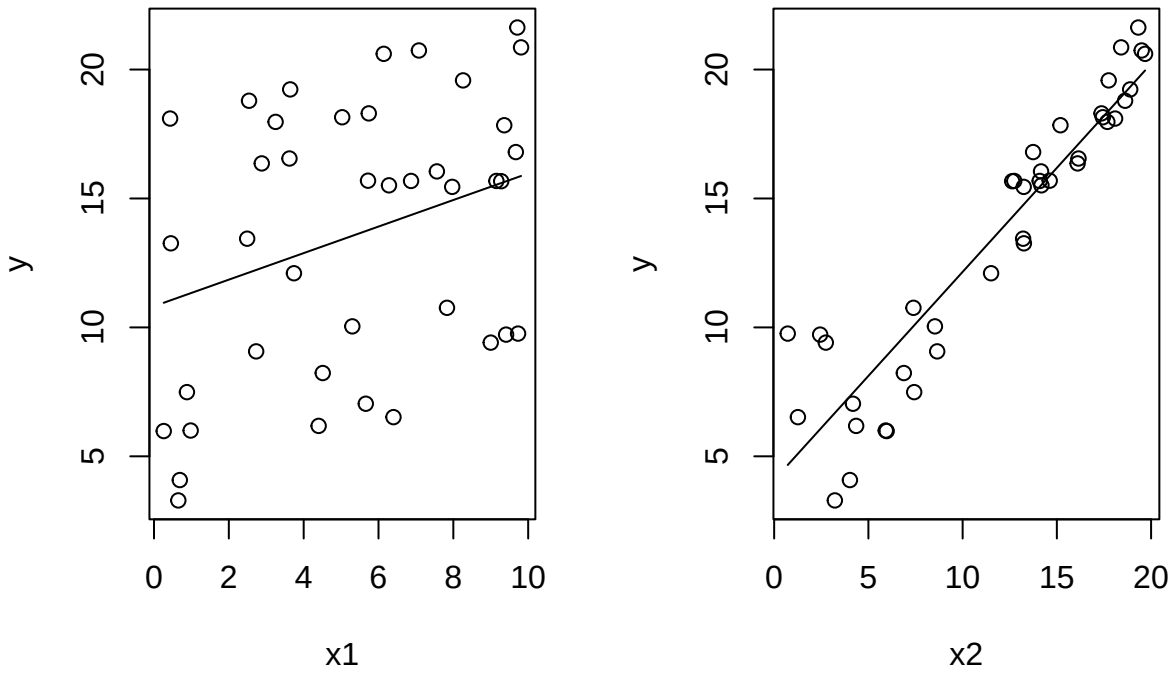


(b)

Display the estimated model graphically as in Figure 10.2

```
par(mfrow = c(1, 2)) #combines graphs into one screen
plot(Pyth[1:40, ]$x1, Pyth[1:40, ]$y,
     xlab = "x1",
     ylab = "y") #plots and labeled axis
curve(coef(LM)[1] + coef(LM)[2] * x + coef(LM)[3] * mean(Pyth[1:40, ]$x2),
```

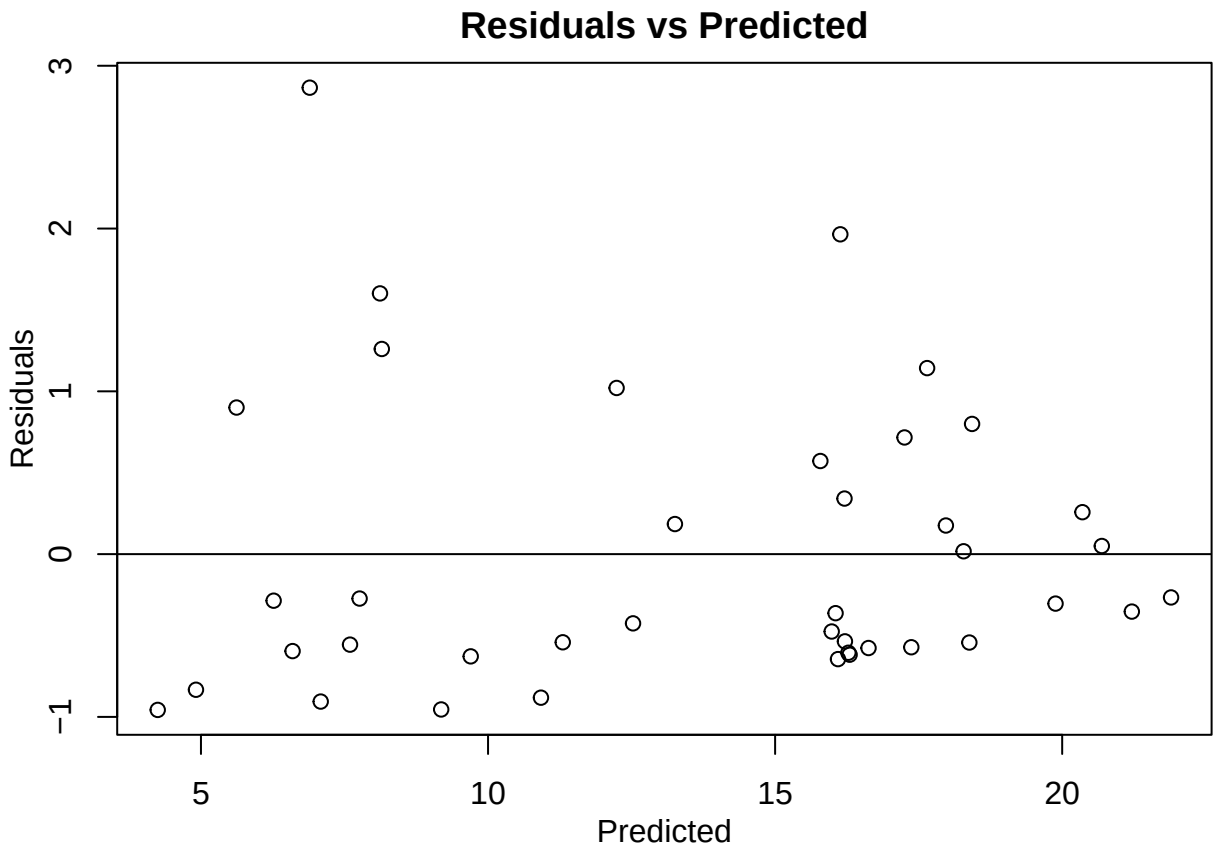
```
      add = TRUE) #regression line
plot(Pyth[1:40, ]$x2, Pyth[1:40, ]$y,
     xlab = "x2",
     ylab = "y") #plots and labeled axis
curve(coef(LM)[1] + coef(LM)[2] * mean(Pyth[1:40, ]$x1) + coef(LM)[3] * x,
      add = TRUE) #regression line
```



(c)

Make a residual plot for this model. Do the assumptions appear to be met?

```
predicted <- predict(stanGlm) #predicted stanGlm
residual <- Pyth$y[1:40] - predicted #residual equation
par(mar = c(3,3,2,1), mgp = c(2, 1, 0)) #gives area to label axis
plot(predicted, residual,
     xlab = "Predicted",
     ylab = "Residuals",
     main = "Residuals vs Predicted") #plot and labeled axis
abline(h = 0) #regression line at y = 0 to see if error assumptions are met
```




```
cat("From the scatterplot, the assumption appears to not be met")
```

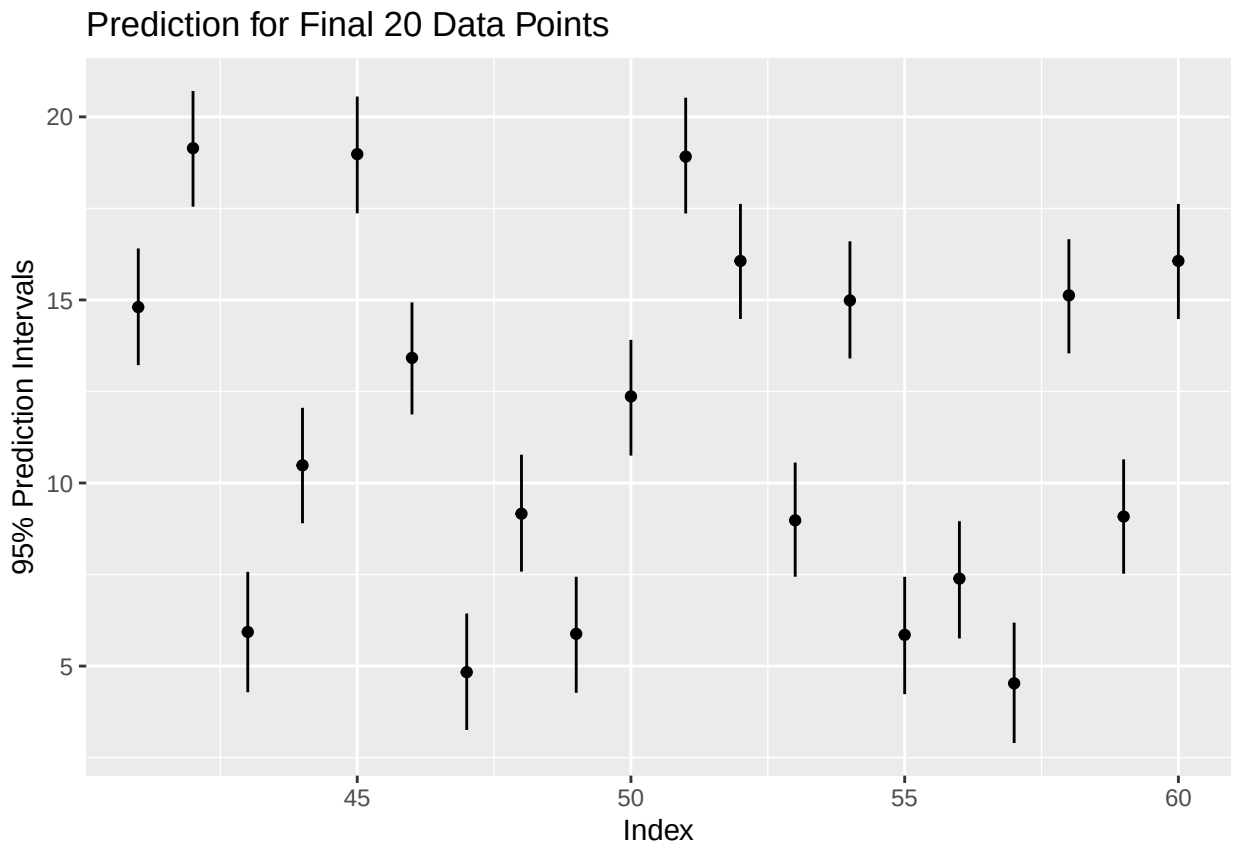
```
## From the scatterplot, the assumption appears to not be met
```

(d)

Make predictions for the remaining 20 data points in the file. How confident do you feel about these predictions?

```
p = posterior_predict(stanGlm, newdata = Pyth[41:60, 2:3])
# looking at remaining values with newdata
p_upper = apply(p, MARGIN = 2, FUN = quantile, probs = 0.95)
# 95% probability
p_lower = apply(p, MARGIN = 2, FUN = quantile, probs = 0.05)
# 1 - 95% probability
p_mean = apply(p, MARGIN = 2, FUN = mean)
# setting mean prediction
pre = data.frame(upper = p_upper, lower = p_lower,
                 point_estimate = p_mean,
                 index = 41:60)
# data frame setup
ggplot(pre) +
  geom_point(aes(index, point_estimate)) +
  geom_segment(aes(index, xend = index,
```

```
                  upper, yend = lower)) +
  labs(x = "Index",
       y = "95% Prediction Intervals",
       title = "Prediction for Final 20 Data Points")
```




```
# plot and labeled axis

cat("Based on the 95% prediction intervals. There seems to be high confidence in the predictions.")
```

```
## Based on the 95% prediction intervals. There seems to be high confidence in the predictions.
```

12.5

Logarithmic transformation and regression: Consider the following regression:

$$\log(\text{weight}) = -3.8 + 2.1 \log(\text{height}) + \text{error},$$

with errors that have standard deviation 0.25. Weights are in pounds and heights are in inches.

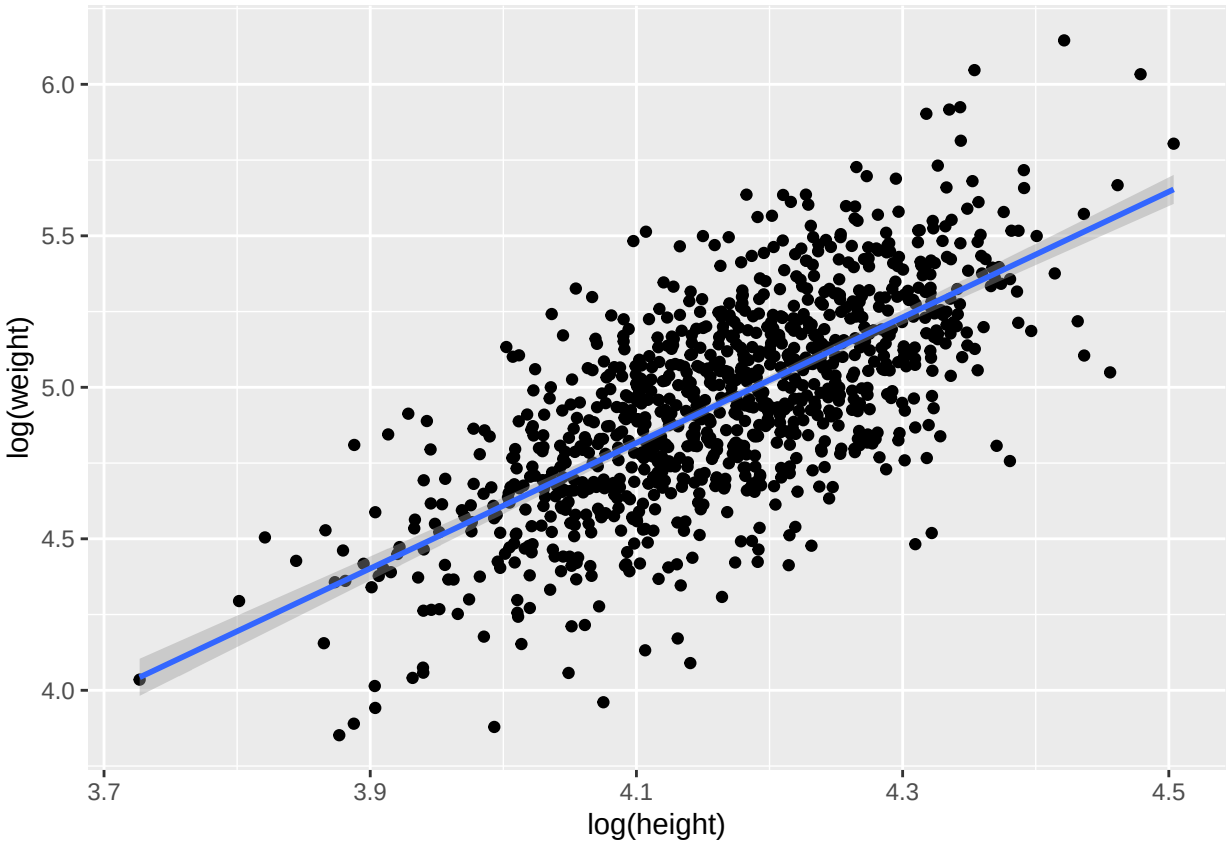
(a)

Fill in the blanks: Approximately 68% of the people will have weights within a factor of 1.284 and 0.25 of their predicted values from the regression.

(b)

Using pen and paper, sketch the regression line and scatterplot of log(weight) versus log(height) that make sense and are consistent with the fitted model. Be sure to label the axes of your graph.

```
height = rnorm(n = 1000, mean = 65, sd = 7)
# Looked up average height in a male 171 and female 159 and averaged it expected a 50/50 population. Al
weight = exp(-3.8 + 2.1 * log(height) + rnorm(n = 1000, mean = 0, sd = 0.25)) #used given values
globals = data.frame(weight = weight, height = height)
ggplot(globals, aes(x = log(height), y = log(weight))) +
  geom_point() +
  geom_smooth(formula = 'y~x', method = "lm") #plot and labeled axis
```



12.6

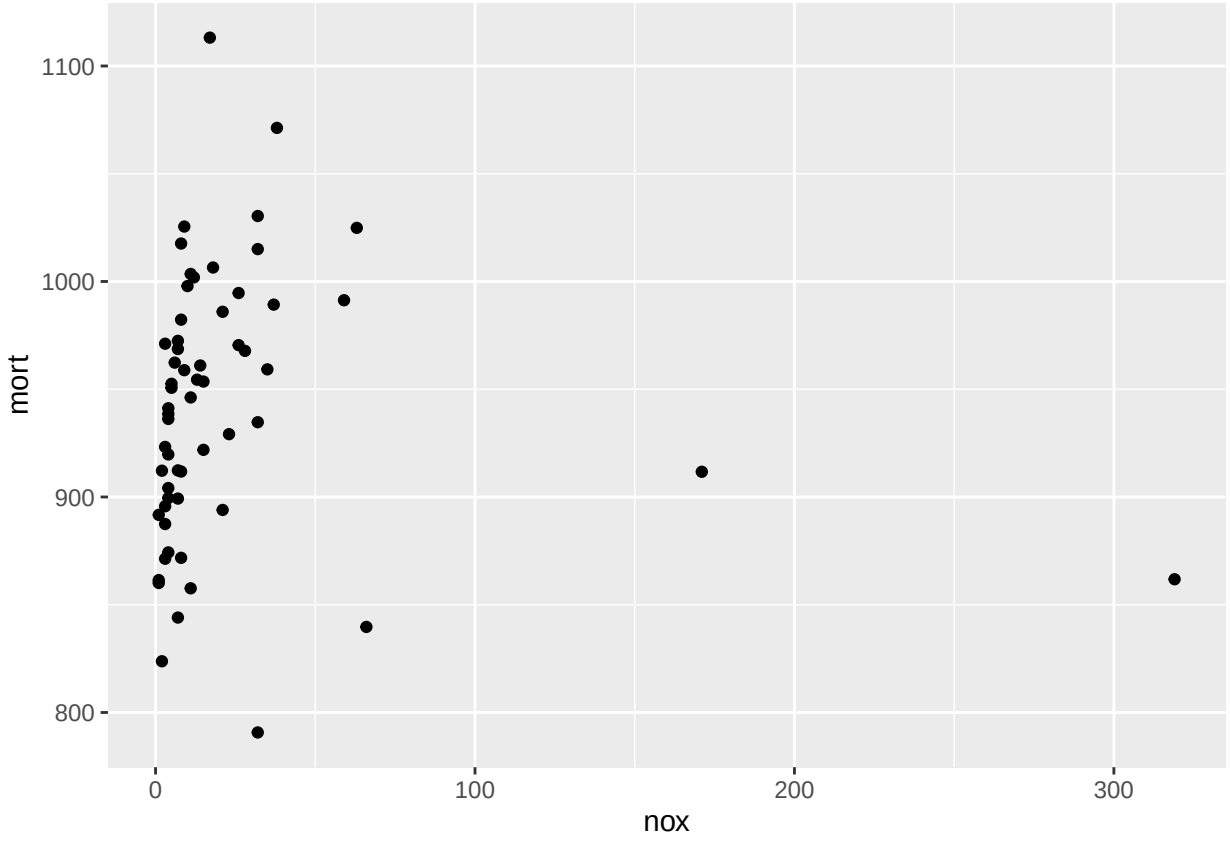
Logarithmic transformations: The folder `Pollution` contains mortality rates and various environmental factors from 60 US metropolitan areas. For this exercise we shall model mortality rate given nitric oxides, sulfur dioxide, and hydrocarbons as inputs. this model is an extreme oversimplication, as it combines all sources of mortality and does not adjust for crucial factors such as age and smoking. We use it to illustrate log transformation in regression.

```
Pollution <- read.csv("pollution.csv", header=TRUE)
# setting pollution.csv from git
```

(a)

Create a scatterplot of mortality rate versus level of nitric oxides. Do you think linear regression will fit these data well? Fit the regression and evaluate a residual plot from the regression.

```
ggplot(data = Pollution) +
  geom_point(aes(nox, mort)) #graph scatterplot
```




```
lm_model = lm(mort ~ nox, Pollution)
summary(lm_model) #summarize the linear model
```

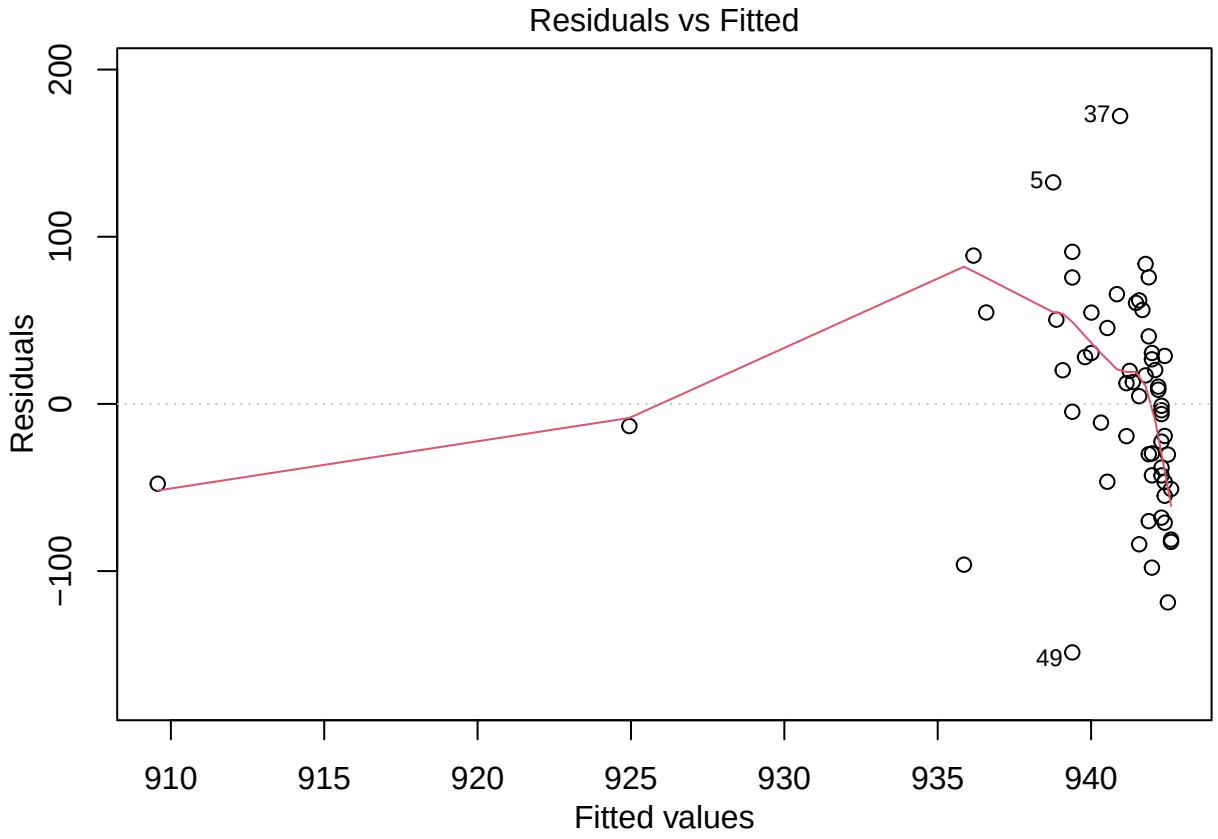
```
##
## Call:
## lm(formula = mort ~ nox, data = Pollution)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -148.654  -43.710    1.751   41.663  172.211
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 942.7115     9.0034 104.706   <2e-16 ***
## nox          -0.1039     0.1758  -0.591    0.557
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 62.55 on 58 degrees of freedom
## Multiple R-squared:  0.005987,   Adjusted R-squared:  -0.01115
## F-statistic: 0.3494 on 1 and 58 DF,  p-value: 0.5568
```

```
cat("Based on the regression model. The linear regression will not fit the data well.")
```

```
## Based on the regression model. The linear regression will not fit the data well.
```

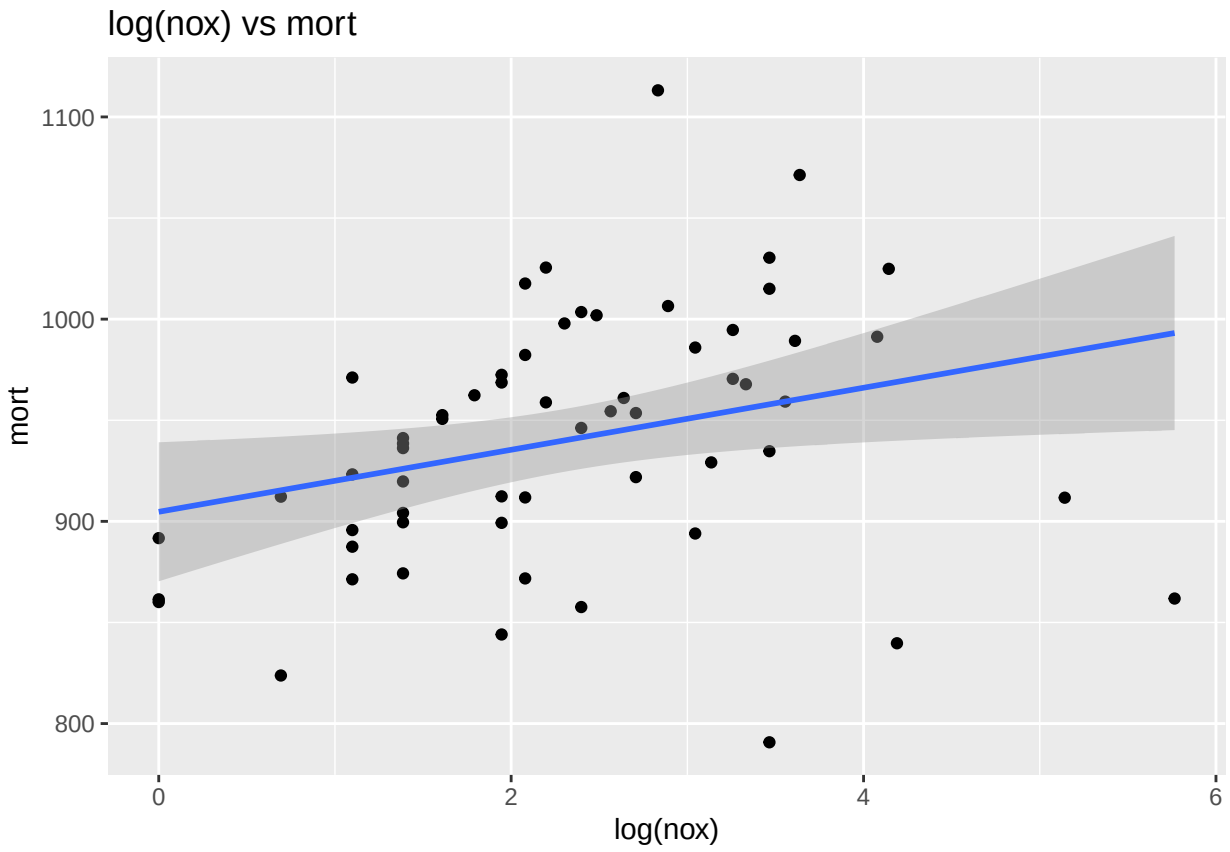
```
par(mar = c(3,3,2,1), mgp = c(2,1,0))
plot(lm_model, which = 1) #evaluated the residual plot from the regression
```



(b)

Find an appropriate reansformation that will result in data more appropriate for linear regression. Fit a regression to the transformed data and evaluate the new residual plot.

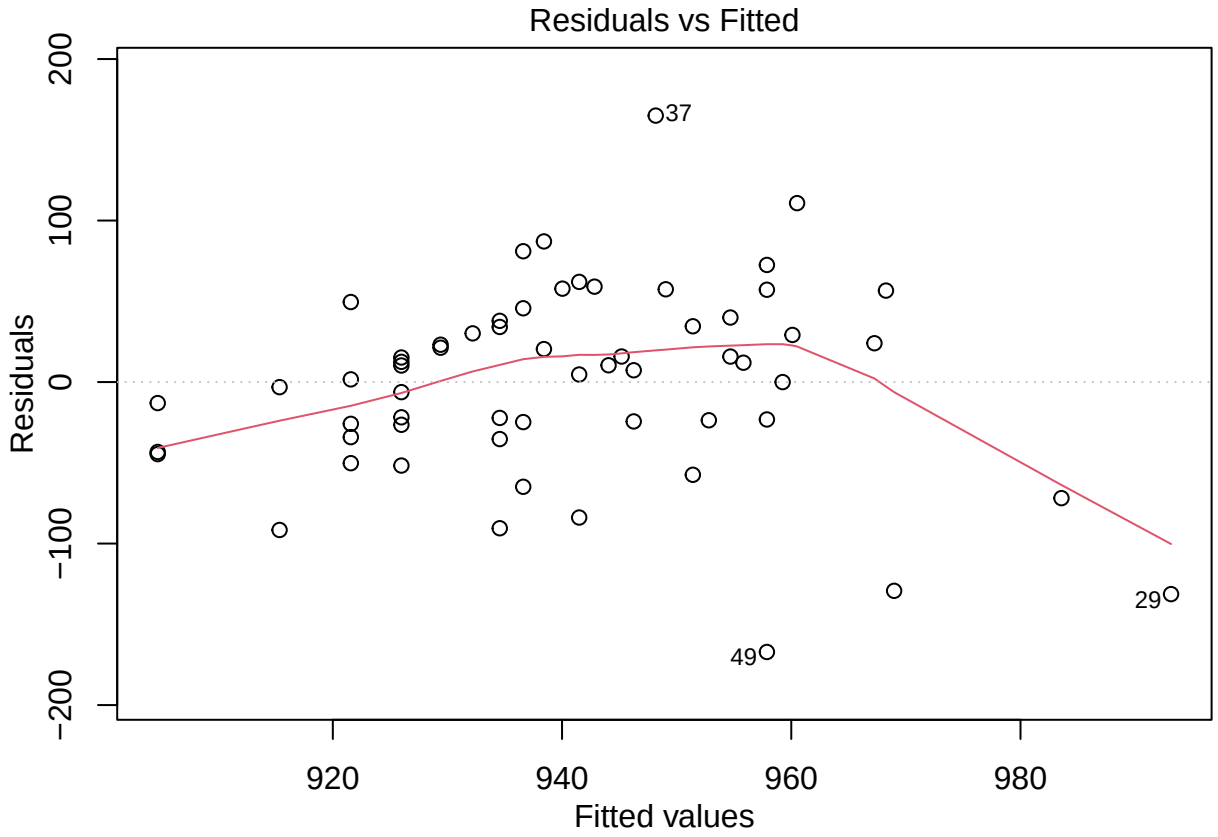
```
# I used log(nox) vs mort because it seemed from 12.6a that nox had to large of outliers that would fit
ggplot(data = Pollution, aes(x = log(nox), y = mort)) +
  geom_point() +
  geom_smooth(formula = "y ~ x", method = "lm") +
  labs(x = "log(nox)",
       y = "mort",
       title = "log(nox) vs mort") #plot and labeled axis
```




```
lm_model = lm(mort ~ log(nox), data = Pollution)
summary(lm_model) #summarize the logarithmic regression model
```

```
##
## Call:
## lm(formula = mort ~ log(nox), data = Pollution)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -167.140  -28.368    8.778   35.377  164.983
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  904.724     17.173  52.684   <2e-16 ***
## log(nox)      15.335      6.596   2.325   0.0236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.01 on 58 degrees of freedom
## Multiple R-squared:  0.08526,    Adjusted R-squared:  0.06949
## F-statistic: 5.406 on 1 and 58 DF,  p-value: 0.02359
```

```
par(mar = c(3,3,2,1), mgp = c(2,1,0))
plot(lm_model, which = 1) #evaluated the residual plot from the regression
```



(c)

Interpret the slope coefficient from the model you chose in (b)

For every 1 increase in log(nitric oxides), on average there is a 15.335 increase in mortality

(d)

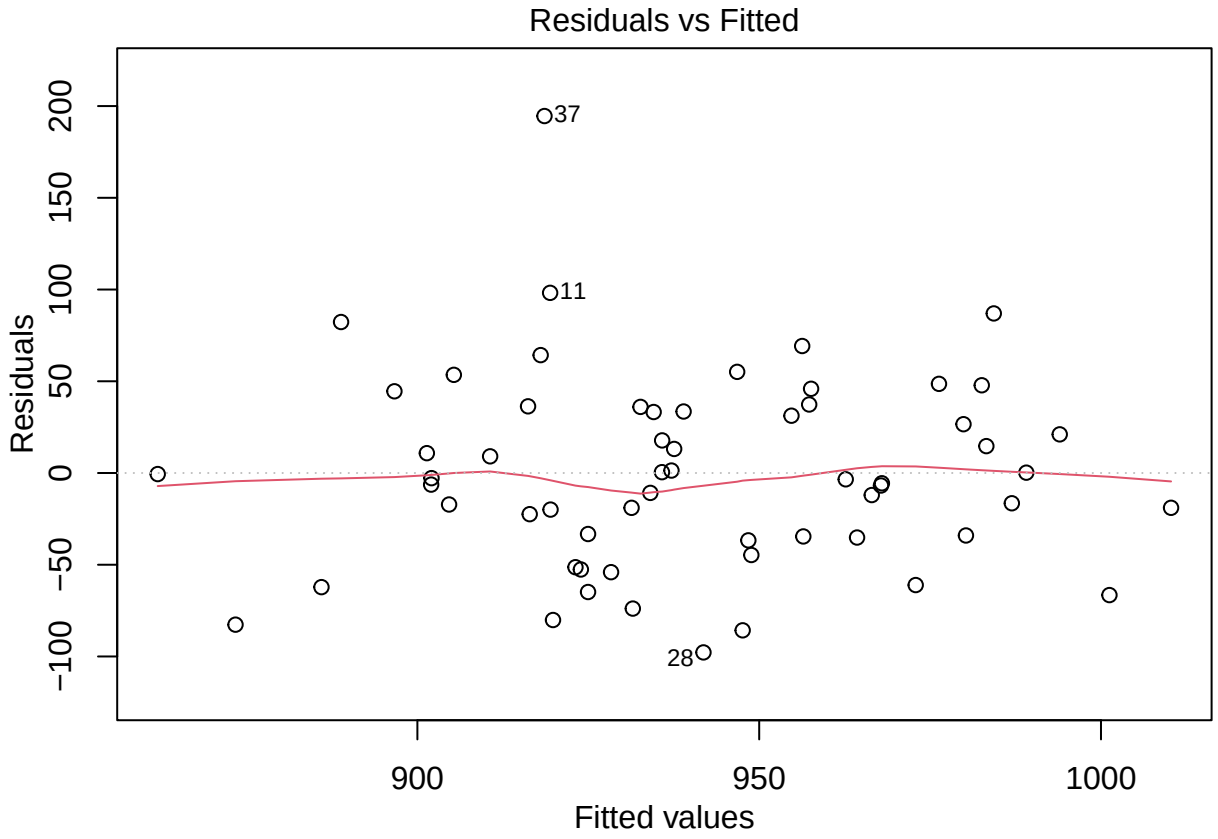
Now fit a model predicting mortality rate using levels of nitric oxides, sulfur dioxide, and hydrocarbons as inputs. Use appropriate transformation when helpful. Plot the fitted regression model and interpret the coefficients.

```
lm_model = lm(mort ~ log(nox) + log(so2) + log(hc), data = Pollution)
summary(lm_model) #summarize the logarithmic regression model
```

```
##
## Call:
## lm(formula = mort ~ log(nox) + log(so2) + log(hc), data = Pollution)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -97.793 -34.728  -3.118  34.148 194.567
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  924.965     21.449  43.125  < 2e-16 ***
## log(nox)      58.336     21.751   2.682  0.00960 **
## log(so2)      11.762      7.165   1.642  0.10629
## log(hc)      -57.300     19.419  -2.951  0.00462 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.36 on 56 degrees of freedom
## Multiple R-squared:  0.2752, Adjusted R-squared:  0.2363
## F-statistic: 7.086 on 3 and 56 DF,  p-value: 0.0004044
```

```
par(mar = c(3,3,2,1), mgp = c(2, 1, 0))
plot(lm_model, which = 1) #plot the fitted regression model
```




```
cat("The coefficients: \n
intercept - when all other values are 1 (since we are dealing with a logarithmic model) our average exp
log(nox) - for every 1 increase in log(nitric oxides), on average there is a 58.336 increase in mortali
log(so2) - for every 1 increase in log(sulfur dioxide), on average there is a 11.762 increase in mortal
log(hc) - for every 1 increase in log(hydrocarbons), on average there is a 57.3 decrease in mortality")
```

```
## The coefficients:
##
## intercept - when all other values are 1 (since we are dealing with a logarithmic model) our average
##
```

```
## log(nox) - for every 1 increase in log(nitric oxides), on average there is a 58.336 increase in morta
##
## log(so2) - for every 1 increase in log(sulfur dioxide), on average there is a 11.762 increase in mor
##
## log(hc) - for every 1 increase in log(hydrocarbons), on average there is a 57.3 decrease in mortalit
```

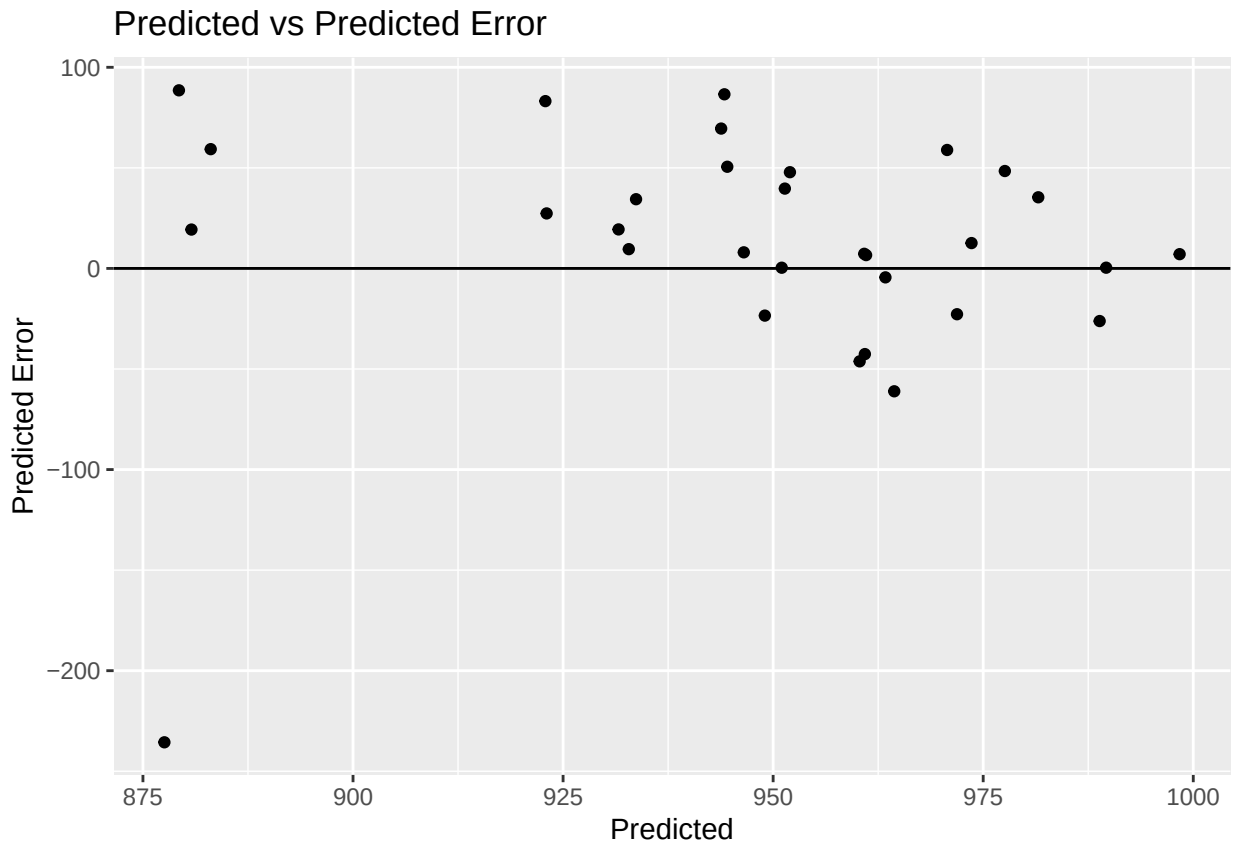
(e)

Cross validate: fit the model you chose above to the first half of the data and then predict for the second half. You used all the data to construct the model in (d), so this is not really cross validation, but it gives a sense of how the steps of cross validation can be implemented.

```
n = dim(Pollution)[1] / 2 #finding first half of data
lm_model = lm(mort ~ log(nox) + log(so2) + log(hc), data = Pollution, subset = 1:n) #linear regression
summary(lm_model) #summarize the logarithmic regression model
```

```
##
## Call:
## lm(formula = mort ~ log(nox) + log(so2) + log(hc), data = Pollution,
##     subset = 1:n)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -110.358  -36.766   -1.032   35.049   82.107
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   899.97      25.71  35.009   <2e-16 ***
## log(nox)       10.57      29.59   0.357   0.7240
## log(so2)       21.87      12.32   1.774   0.0877 .
## log(hc)       -17.47      26.21  -0.667   0.5108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.07 on 26 degrees of freedom
## Multiple R-squared:  0.2522, Adjusted R-squared:  0.1659
## F-statistic: 2.922 on 3 and 26 DF,  p-value: 0.05277
```

```
plm_model = predict(lm_model,
                    newdata = Pollution[n:(2 * n), ]) #second half data
ggplot() +
  geom_point(aes(plm_model, plm_model -
                   Pollution$mort[n:(2 * n)])) +
  geom_hline(yintercept = 0) +
  labs(x = "Predicted",
       y = "Predicted Error",
       title = "Predicted vs Predicted Error") #plot and labeled axis
```



12.7

Cross validation comparison of models with different transformations of outcomes: when we compare models with transformed continuous outcomes, we must take into account how the nonlinear transformation warps the continuous outcomes. Follow the procedure used to compare models for the mesquite bushes example on page 202.

(a)

Compare models for earnings and for log(earnings) given height and sex as shown in page 84 and 192. Use `earnk` and `log(earnk)` as outcomes.

```
Earnings <- read.csv("earnings.csv")
Earnings$log_earnk <- log(Earnings$earnk)

earnings <- na.omit(Earnings) #getting rid of NA
earnings <- earnings[!is.infinite(earnings$log_earnk),]

regEarnK <- stan_glm(earnk ~ height + male, data = earnings, refresh = 0)
logEarnK <- stan_glm(log_earnk ~ height + male, data = earnings, refresh = 0)
#Reporting these Bayesian regression into models

regEarnK
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      earnk ~ height + male
##  observations: 1295
##  predictors:   3
## ------
##             Median MAD_SD
## (Intercept) -13.8   15.0
## height        0.5    0.2
## male         10.3    1.8
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 23.0    0.4
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
logEarnK
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log_earnk ~ height + male
##  observations: 1295
##  predictors:   3
## ------
##             Median MAD_SD
## (Intercept) 1.6    0.6
## height      0.0    0.0
## male        0.4    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

(b)

Compare models from other exercises in this chapter.

Comparing the models from other exercises, for the log(earnK), it seems as if it is similar to the other models in a sense that they are very centered and stable due to the very low standard deviation (the normal distribution). Even for the regular earnK model, we can see that the standard deviation is also really low resulting in similar results as it's log counterpart.

12.8

Log-log transformations: Suppose that, for a certain population of animals, we can predict log weight from log height as follows:

- An animal that is 50 centimeters tall is predicted to weigh 10 kg.
- Every increase of 1% in height corresponds to a predicted increase of 2% in weight.
- The weights of approximately 95% of the animals fall within a factor of 1.1 of predicted values.

(a)

Give the equation of the regression line and the residual standard deviation of the regression.

Since log(weight) = 0.02 * log(height) + (log(10) - 2log(50)) The equation of the regression line is: log(weight) = 0.02 * log(height) - 5.52 + error

For the residual standard deviation: 0.0477

(b)

Suppose the standard deviation of log weights is 20% in this population. What, then, is the R^2 of the regression model described here?

For R^2, we have: 1 - $(0.477^{2)/(0.2)}2 = 0.9431$

12.9

Linear and logarithmic transformations: For a study of congressional elections, you would like a measure of the relative amount of money raised by each of the two major-party candidates in each district. Suppose that you know the amount of money raised by each candidate; label these dollar values D_i and R_i. You would like to combine these into a single variable that can be included as an input variable into a model predicting vote share for the Democrats. Discuss the advantages and disadvantages of the following measures:

(a)

The simple difference, $D_i - R_i$

Advantage: When centered (Di = Ri), the visualization and the output is very easy to interpret.

Disadvantage: When it comes to a single output, negative numbers can be an issue with interpreting the single variable; in addition, even with absolute values of these differences will only confuse which party has the more money raised than the other.

(b)

The ratio, D_i/R_i

Advantage: There might be times where we need to see the percent difference between the two values and we can see how much more funding percentage one party received compared to the other party.

Disadvantage: When dealing with ratios, when they are centered the value will be at 1 instead of 0 since a number/number = 1 and this can cause confusion due to it being asymmetric.

(c)

The difference on the logarithmic scale, $\log D_i - \log R_i$

Advantage: This can show the differences in percentage of funding each party received, which can be good to know in certain statistics

Disadvantage: The log graphs are usually easier to graph in certain aspects of exponentially increasing values but harder to interpret when it comes to explaining it's visualization.

(d)

The relative proportion, $D_i/(D_i + R_i)$.

Advantage: This can show important information to those who wants to see the percentage of funding for a specific party in comparison to the whole, which can give the reader a good description of the bigger picture.

Disadvantage: Just like 12.9b, the center will be in this case at 0.5 instead of 0 and this can cause confusion. If the values of Di and Ri are both 0, then there will be an error in the equation.

12.11

Elasticity: An economist runs a regression examining the relations between the average price of cigarettes, P, and the quantity purchased, Q, across a large sample of counties in the United States, assuming the functional form, $\log Q = \alpha + \beta \log P$. Suppose the estimate for β is 0.3. Interpret this coefficient.

Since we are dealing with a log function, for every 1% increase we have in cigarette prices, on average we see a 0.3% increase in the quantity purchased.

12.13

Building regression models: Return to the teaching evaluations data from Exercise 10.6. Fit regression models predicting evaluations given many of the inputs in the dataset. Consider interactions, combinations of predictors, and transformations, as appropriate. Consider several models, discuss in detail the final model that you choose, and also explain why you chose it rather than the others you had considered.

```
Beauty <- read.csv("beauty.csv", header = TRUE)

beauty1 <- stan_glm(eval ~ beauty, data = Beauty, refresh = 0)
Beauty$prediction1 = predict(beauty1, newdata = Beauty)
Beauty$minority <- as.factor(Beauty$minority)
beauty2 <- stan_glm(eval ~ beauty + minority, data = Beauty, refresh = 0)
Beauty$prediction2 = predict(beauty2, newdata = Beauty)
beauty3 <- stan_glm(eval ~ beauty + minority + beauty * minority, data = Beauty, refresh = 0)
Beauty$prediction3 = predict(beauty3, newdata = Beauty)
# Set up all the new columns as predictions1-3

mean_eval <- mean(Beauty$eval)
mean1 <- mean(Beauty$prediction1) - mean_eval
mean2 <- mean(Beauty$prediction2) - mean_eval
mean3 <- mean(Beauty$prediction3) - mean_eval
# Used the predictions to find the difference in means from the actual mean of the evaluation

mean1
```

```
## [1] 0.0004108208
```

```
mean2
```

```
## [1] -0.0002379559
```

```
mean3
```

```
## [1] -0.0004457735
```

```
summary(beauty1)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 463
##  predictors:   2
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) 4.0    0.0  4.0   4.0   4.0
## beauty      0.1    0.0  0.1   0.1   0.2
## sigma       0.5    0.0  0.5   0.5   0.6
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 4.0    0.0  4.0   4.0   4.0
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  3669
## beauty        0.0  1.0  3443
## sigma         0.0  1.0  3684
## mean_PPD      0.0  1.0  4050
## log-posterior 0.0  1.0  1802
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
summary(beauty2)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
```

```
##  formula:      eval ~ beauty + minority
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 463
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept)  4.0    0.0  4.0   4.0   4.1
## beauty       0.1    0.0  0.1   0.1   0.2
## minority1   -0.1    0.1 -0.2  -0.1   0.0
## sigma        0.5    0.0  0.5   0.5   0.6
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 4.0    0.0  4.0   4.0   4.0
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  5425
## beauty        0.0  1.0  5361
## minority1     0.0  1.0  5602
## sigma         0.0  1.0  5429
## mean_PPD      0.0  1.0  4178
## log-posterior 0.0  1.0  1680
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
summary(beauty3)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty + minority + beauty * minority
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 463
##  predictors:   4
##
## Estimates:
##                    mean   sd   10%   50%   90%
## (Intercept)       4.0    0.0  4.0   4.0   4.1
## beauty            0.2    0.0  0.1   0.2   0.2
## minority1        -0.1    0.1 -0.2  -0.1   0.0
## beauty:minority1 -0.2    0.1 -0.4  -0.2  -0.1
## sigma             0.5    0.0  0.5   0.5   0.6
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
```

```
## mean_PPD 4.0    0.0  4.0   4.0   4.0
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                  mcse Rhat n_eff
## (Intercept)      0.0  1.0  4892
## beauty           0.0  1.0  4464
## minority1        0.0  1.0  5388
## beauty:minority1 0.0  1.0  4615
## sigma            0.0  1.0  5589
## mean_PPD         0.0  1.0  4336
## log-posterior    0.0  1.0  1766
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

12.14

Prediction from a fitted regression: Consider one of the fitted models for mesquite leaves, for example `fit_4`, in Section 12.6. Suppose you wish to use this model to make inferences about the average mesquite yield in a new set of trees whose predictors are in data frame called new_trees. Give R code to obtain an estimate and standard error for this population average. You do not need to make the prediction; just give the code.

```
# mesquiteData <- "https://raw.githubusercontent.com/avehtari/ROS-Examples/refs/heads/master/"
# mesquite <- read.table(paste0(mesquiteData, "Mesquite/data/mesquite.dat"), header = TRUE)
#  mesquite$cvolume <- mesquite$diam1 * mesquite$diam2 * mesquite$canopy_height
#  mesquite$carea <- mesquite$diam1 * mesquite$diam2
#  mesquite$cshape <- mesquite$diam1 / mesquite$diam2
#  # The new columns that was created into the mesquite data
#
#  fit_4 <- stan_glm(formula = log(weight) ~ log(cvolume) +
#                      log(carea) + log(cshape) +
#                      log(total_height) + log(density) + group,
#                    data = mesquite, refresh = 0) #set the fitted models into fit_4
#  p = posterior_predict(fit_4, newdata = new_trees, fun = exp)
#  # Code will NOT run because we do not have the new_trees data frame just making an inference
#  pavg = apply(p, MARGIN = 2, FUN = mean)
#  popavg = mean(pavg) #computed population average
#  popmsd = sd(pmean) #computed standard error
#
#  # This is the code when the new_trees data.frame would have loaded
```