Addressing Model Problems

Before we begin, let's load a couple libraries.

```
library(tidyverse)  # library with several helpful tools
library(car)        # this let's us make the nice qqPlots
```

In this tutorial I will work through several of the examples that you read about in Chapter 9 in the Penn State e-book on regression. The reading talked about the intuition of when to use the transformations, so here I'll focus on using `R` to perform them.

Rule of Thumb

Before we begin, keep in minde the following general rules when to trying to fix problems with model assumptions:

- To fix problems with non-linear relationships try transforming the affected predictor variables
- To fix problems with error terms that are non-linear, non-normal, or have non-constant variance try transforming the response variable.

We'll use these rules of thumb for some examples.

Transforming the Predictor Variable

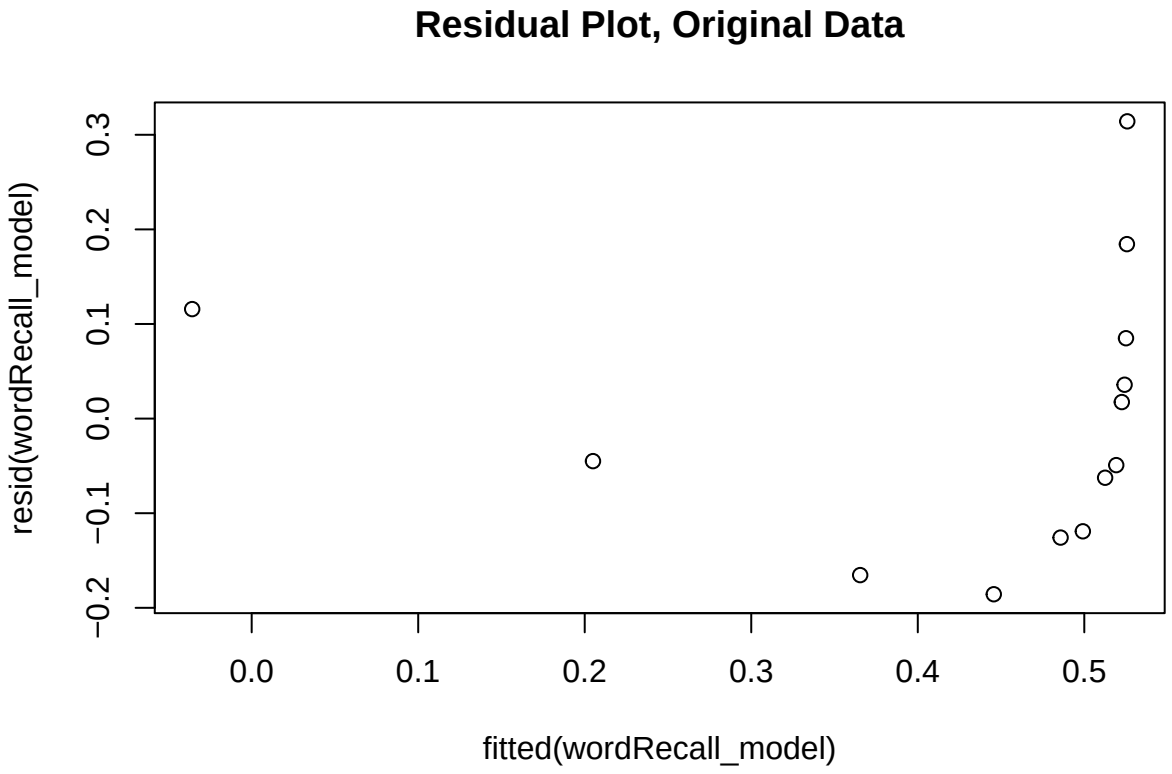
With the Word Recall data from Example 9-1 the authors concluded that the data had a non-linear relationship, but that was the only regression assumption which was not met. To address this problem, they chose to apply a `log` transformation to the predictor variable.

First, let's load the dataset and then recreate the initial linear model, residual plot, and normal probability plot from the text.

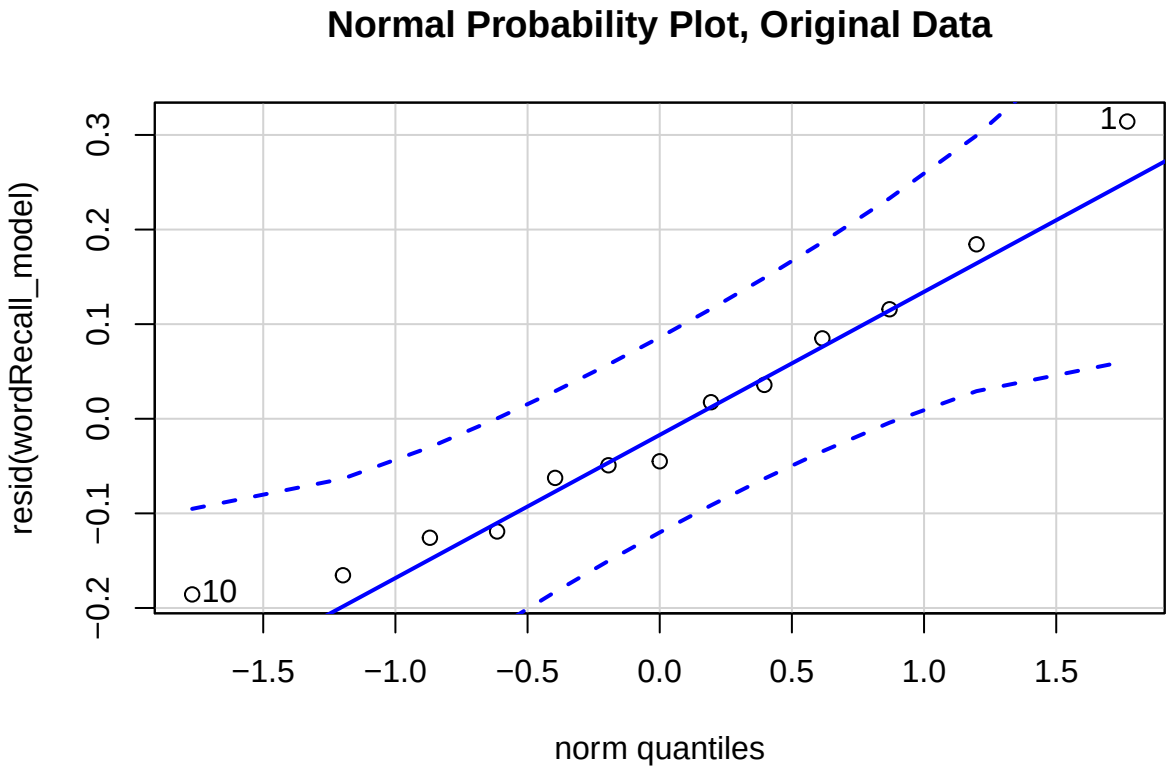
```
wordRecall <- read_csv("wordrecall.csv")
```

```
Parsed with column specification:
cols(
  time = col_double(),
  prop = col_double()
)
```

```
attach(wordRecall)
wordRecall_model <- lm(prop ~ time)
plot(resid(wordRecall_model) ~ fitted(wordRecall_model), main="Residual Plot, Original Data")
```




```
qqPlot(resid(wordRecall_model), main="Normal Probability Plot, Original Data")
```

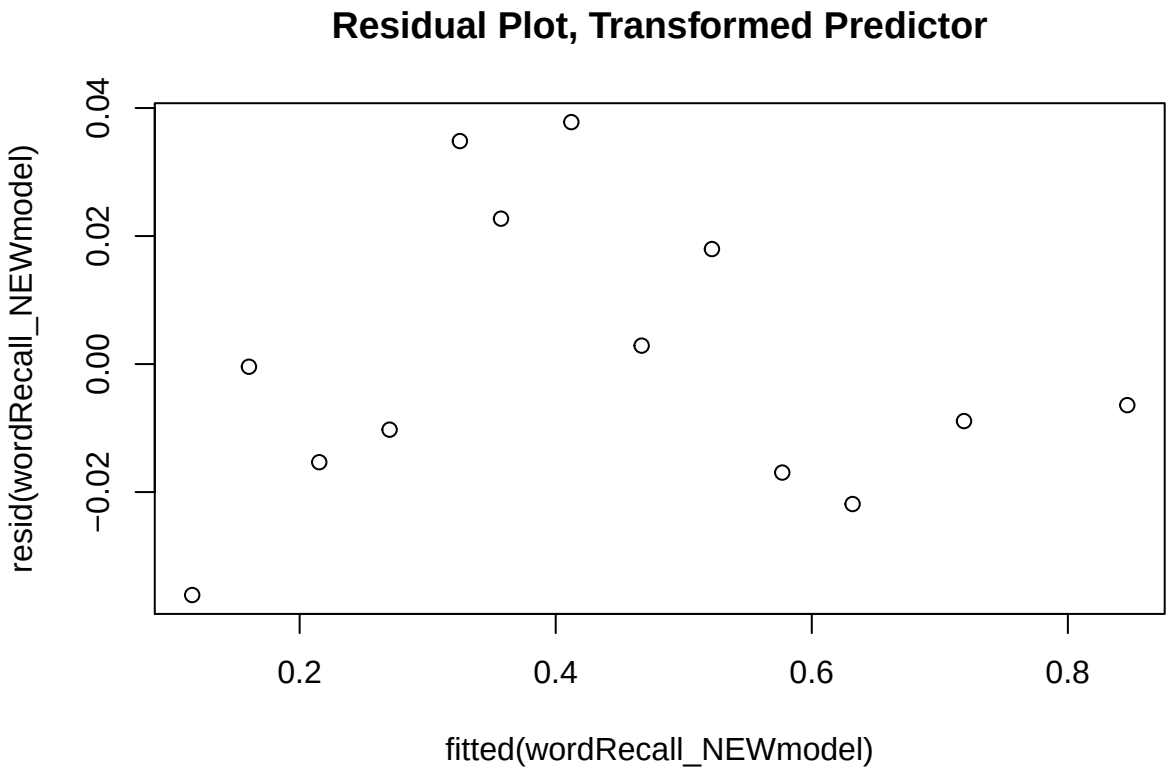



```
[1]  1 10
```

Just as we saw in the reading, the residual plot is not good because it shows a clear non-linear pattern. The normal probability plot is fine indicating the errors are normally distributed.

Ok, so we identified the problem, now to fix it we saw in the text that we needed to apply a `log` transformation to the predictor variable, `time`. Note that `R` uses the function `log` for the natural logarithm that was discussed in the text. So we create a new data vector containing the transformed data, then create a new linear model.

```
lntime <- log(time) # natural log of time
wordRecall_NEWmodel <- lm(prop ~ lntime)
plot(resid(wordRecall_NEWmodel) ~ fitted(wordRecall_NEWmodel), main="Residual Plot, Transformed Predict
```



The residual plot is improved as we expected.

To use this model, let's see what the coefficients are.

```
coefficients(wordRecall_NEWmodel)
```

```
(Intercept)      lntime
 0.84641541 -0.07922691
```

Keep in mind, the equation from this model is

$$prop = 0.846 - 0.079 * \ln(time)$$

So if we want to predict the proportion of words recalled after 500 minutes we need to evaluate:

$$prop = 0.846 - 0.079 * \ln(500)$$

You can use a calculator to evaluate this, or `R` can do this for us (as long as we remember that the natural log in `R` is called `log`):

```
0.846 - 0.079*log(500)
```

```
[1] 0.355046
```

So we predict the proportion of recalled words will be 0.355, or about 36%.

Transforming the Response Variable

With the Mammal Gestation data in Example 9-2 the authors conclude the only problem with the data is unequal variances in the errors. They decide to transform the response variable to address this problem.

Let's load the data, apply a `log` transformation to the response variable, then generate the new model.

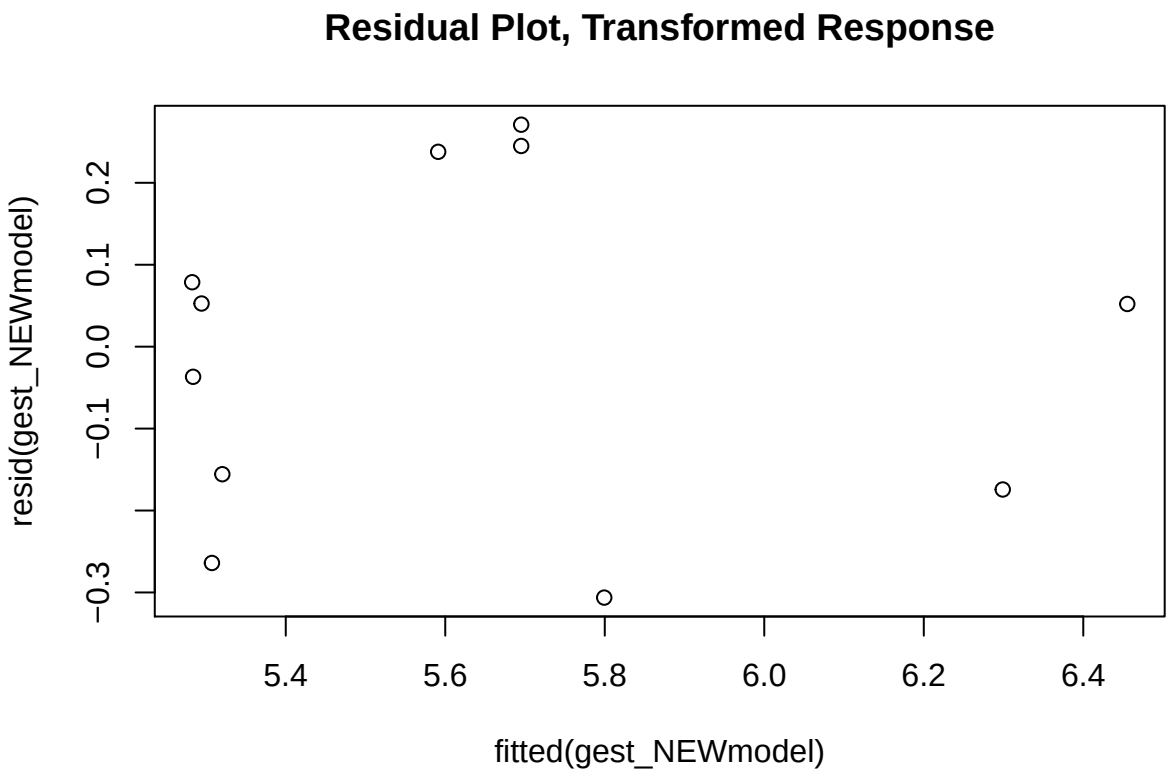
```
gest <- read_csv("mammgest.csv")
```

```
Parsed with column specification:
cols(
  Mammal = col_character(),
  Birthwgt = col_double(),
  Gestation = col_double()
)
```

```
attach(gest)
lnGestation <- log(Gestation)
gest_NEWmodel <- lm( lnGestation ~ Birthwgt)
```

And we can verify what you saw in the reading that the spread of residuals is much better.

```
plot(resid(gest_NEWmodel) ~ fitted(gest_NEWmodel), main="Residual Plot, Transformed Response")
```



Transforming Both

Let's use the Short Leaf data from Example 9-3 in your reading. The authors found problems with non-linearity (implying we should transform the predictor) and with the residuals (implying we should transform the response). Below we load the data, transform both variables, then generate the new model and a residual plot that should look familiar from the text.

```
leaf <- read_csv("shortleaf.csv")
```

```
Parsed with column specification:
cols(
  Diam = col_double(),
  Vol = col_double()
)
```

```
lnDiam <- log(leaf$Diam)
lnVol <- log(leaf$Vol)
leaf_NEWmodel <- lm(lnVol ~ lnDiam)
plot(resid(leaf_NEWmodel) ~ fitted(leaf_NEWmodel), main="Residual Plot, Transformed Response and Predic
```

Let's find the coefficients.

```
coefficients(leaf_NEWmodel)
```

```
(Intercept)      lnDiam
  -2.871790    2.564416
```

Now be careful when using this model. Remember you transformed both variables, so the model is:

$$\ln(Vol) = -2.872 + 2.564 * \ln(Diam)$$

This means that if we want to predict leaf volume for a diameter of 12 we can find:

$$\ln(Vol) = -2.872 + 2.564 * \ln(12)$$

If we compute the right-hand side using `R` or a calculator we find:

$$\ln(Vol) = 3.499$$

Finally we can find the volume estimate by exponentiating both sides (to eliminate the ln).

$$Vol = e^{3.499} = 33.082$$

EPA Data

Let's try one more example that was not in your reading.

Recall in a previous tutorial on checking assumptions we used "EPA gasoline rating 2019.csv"[1] which contains data about 2019 model year vehicles collected by the Enivronmental Protection Agency along with the EPA miles per gallon fuel efficiency ration. This data set includes gasoline powered vehicles only.

We created a model which was significant, but we found that it failed to meet several of our assumptions. Let's try to improve it with data transformations.

```
epa <- read_csv("EPA gasoline rating 2019.csv")
```

```
Parsed with column specification:
cols(
  Model = col_character(),
  Displ = col_double(),
  Cyl = col_double(),
  Trans = col_character(),
  Drive = col_character(),
  `Cert Region` = col_character(),
  Stnd = col_character(),
  `Stnd Description` = col_character(),
  `Underhood ID` = col_character(),
  `Veh Class` = col_character(),
  `Air Pollution Score` = col_double(),
  `City MPG` = col_double(),
  `Hwy MPG` = col_double(),
  `Cmb MPG` = col_double(),
  `Greenhouse Gas Score` = col_double(),
  SmartWay = col_character(),
  `Comb CO2` = col_double()
)
```

```
attach(epa)
epaModel <- lm(`Cmb MPG` ~ Displ + `Veh Class`)
summary(epaModel)
```

```

Call:
lm(formula = `Cmb MPG` ~ Displ + `Veh Class`)

Residuals:
    Min      1Q  Median      3Q     Max
-8.1004 -2.3791 -0.4322  1.2878 29.5314

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               33.44658    0.35288  94.781  < 2e-16 ***
Displ                     -3.11126    0.07015 -44.354  < 2e-16 ***
`Veh Class`midsize car     2.20816    0.33655   6.561 6.52e-11 ***
`Veh Class`minivan        -1.06978    1.04282  -1.026 0.305066
`Veh Class`pickup         -2.17479    0.39483  -5.508 4.01e-08 ***
`Veh Class`small car      -0.45678    0.30189  -1.513 0.130387
`Veh Class`small SUV      -1.84497    0.33102  -5.574 2.77e-08 ***
```

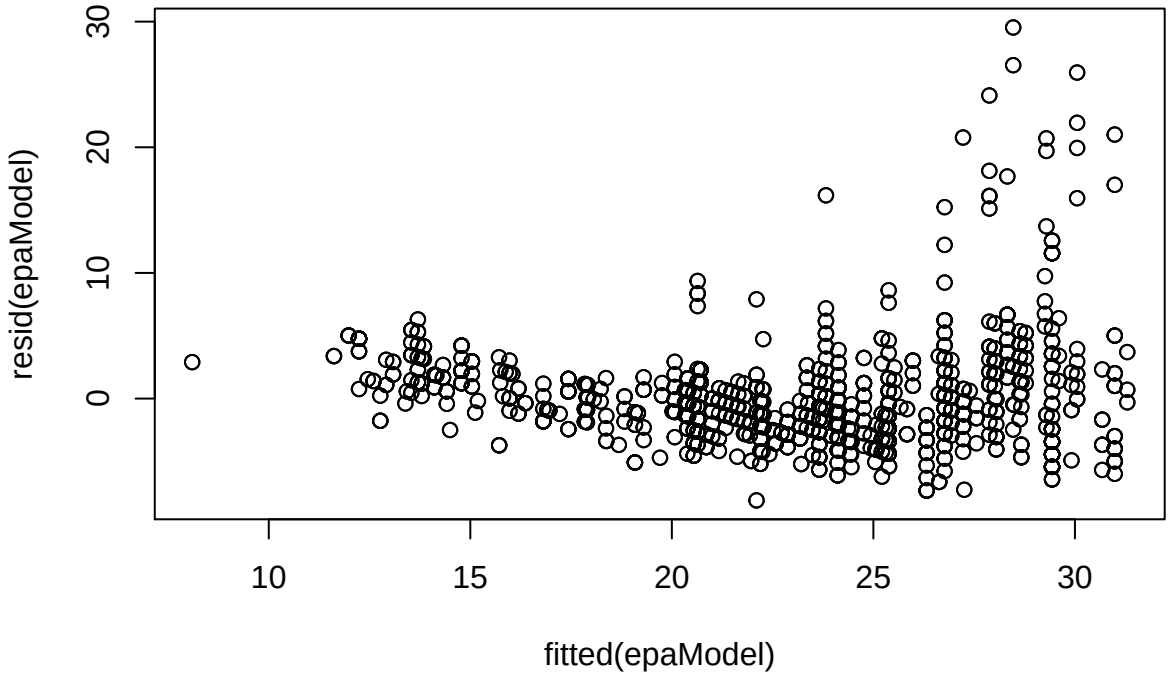
[1] EPA (2019). Fuel Economy Data Set [Data File]. Accessed at https://www.fueleconomy.gov/feg/download.shtml

```
`Veh Class`special purpose -3.87295    0.59900  -6.466 1.22e-10 ***
`Veh Class`standard SUV    -1.91775    0.37379  -5.131 3.12e-07 ***
`Veh Class`station wagon    0.82363    0.48700   1.691 0.090923 .
`Veh Class`van             -6.74606    1.97023  -3.424 0.000627 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.904 on 2404 degrees of freedom
Multiple R-squared:  0.5748,    Adjusted R-squared:  0.5731
F-statistic:   325 on 10 and 2404 DF,  p-value: < 2.2e-16
```

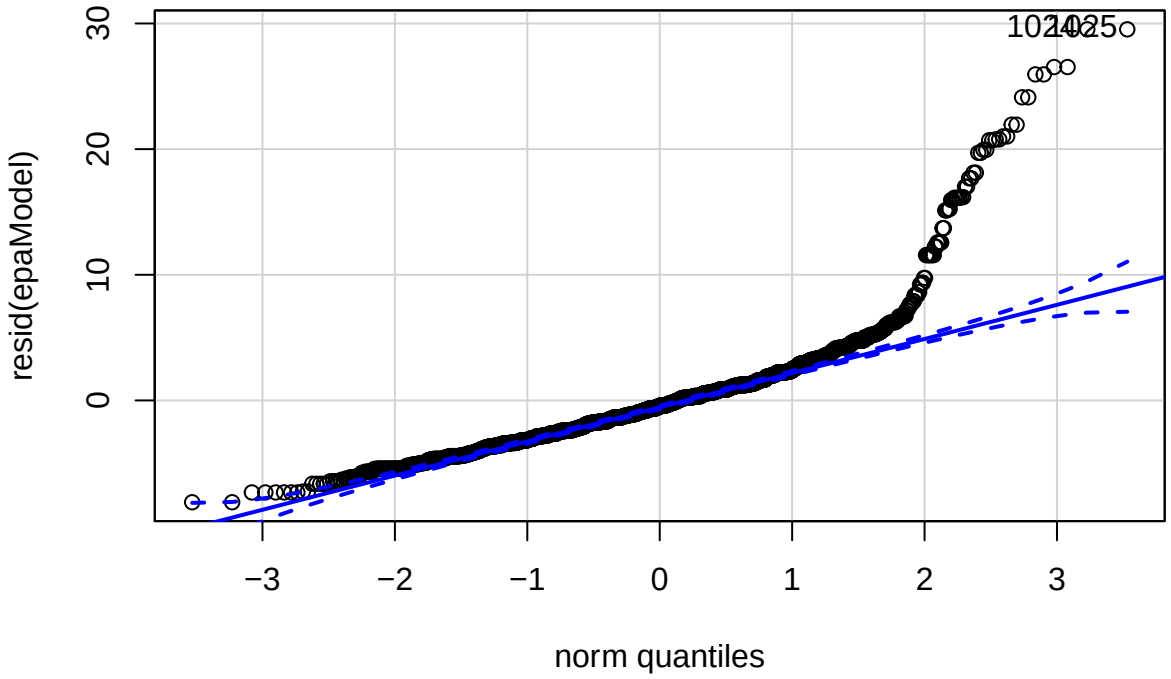
Recall our residual vs fitted plot showed a clear funnel shape indicating non-constant errors.

```
plot(resid(epaModel) ~ fitted(epaModel))
```



And our normal probability plot showed the residuals were likely not normal.

```
qqPlot(resid(epaModel))
```

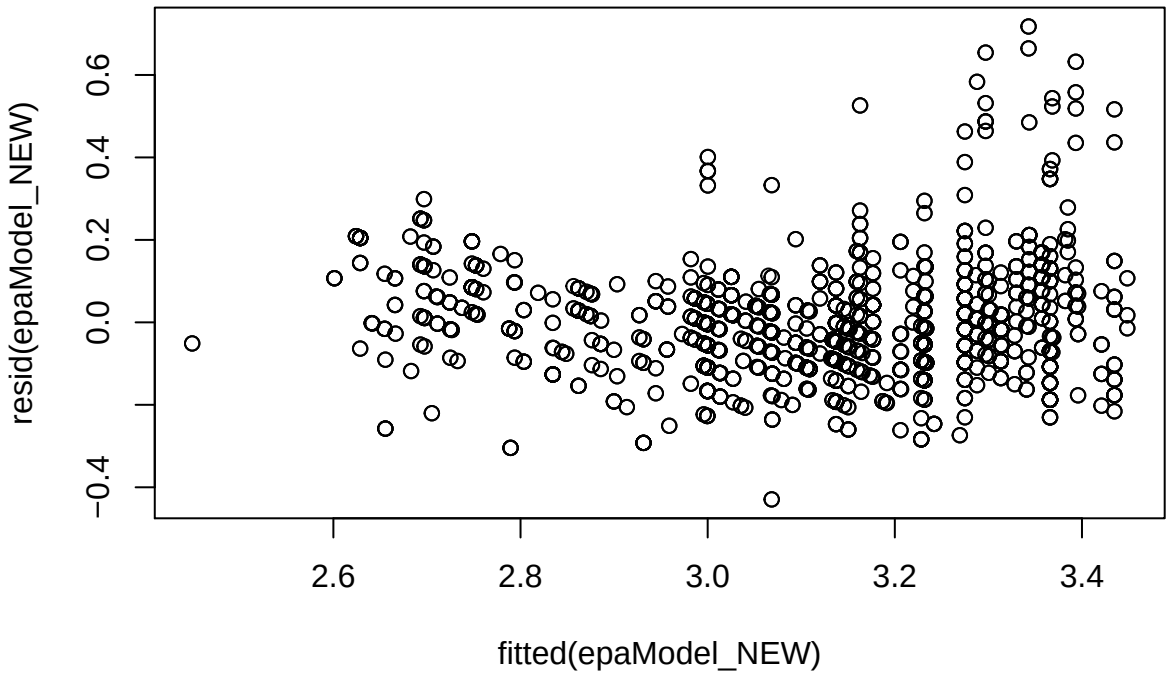



```
[1] 1024 1025
```

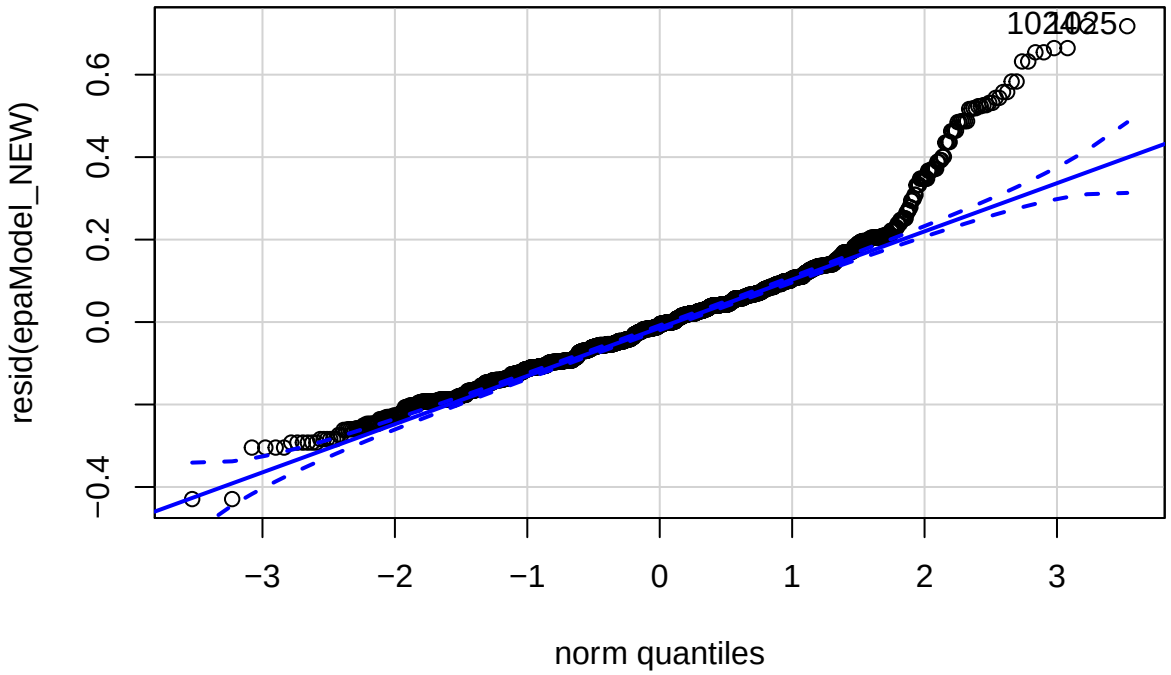
Recall our general guidelines above that when the variances are unequal and the errors are not normal we think to transform the response variable (`Cmb MPG` in this case).

We will try a natural log transformation. We'll use the `log` function to transform the column, then create a new model and the residual plots from it.

```
lnCmbMPG <- log(`Cmb MPG`)
epaModel_NEW <- lm(lnCmbMPG ~ Displ + `Veh Class`)
plot(resid(epaModel_NEW) ~ fitted(epaModel_NEW))
```



```
qqPlot(resid(epaModel_NEW))
```



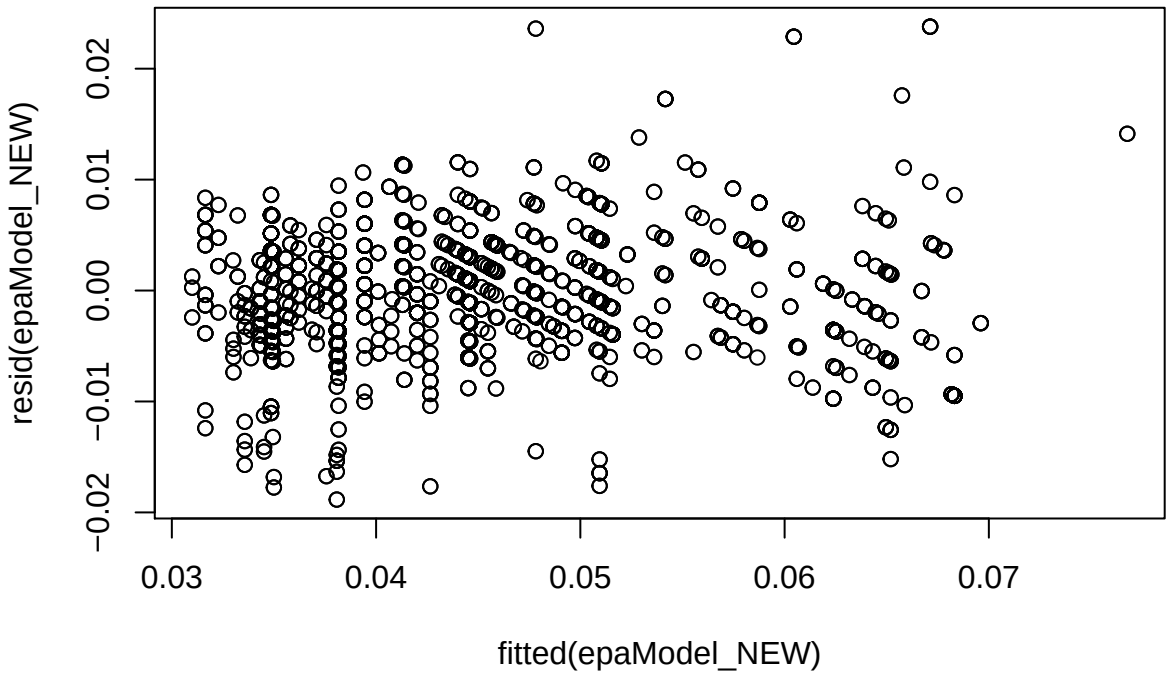

```
[1] 1024 1025
```

The transformation did improve the heteroscedasticity as you can see in the residual vs fitted plot. That plot is generally good.

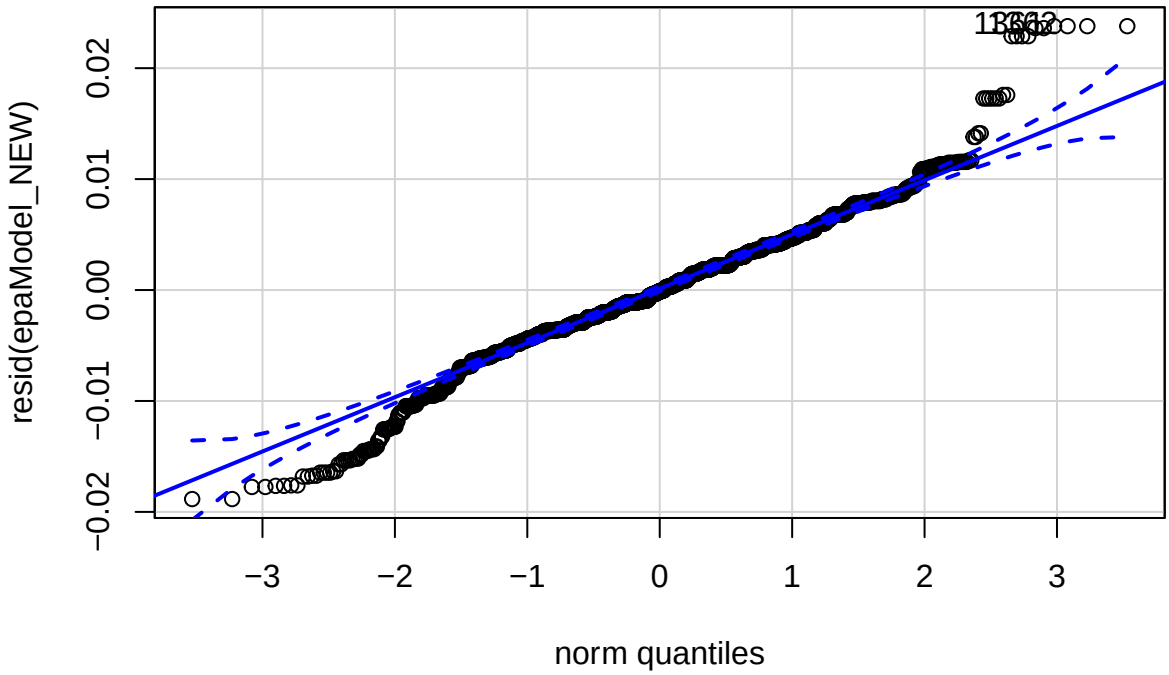
However, there are still normality problems. The normal probability plot still shows significant deviation from the confidence band.

We can try other transformations. For instance, below I tried y^{-1}.

```
transformedCmbMPG <- (`Cmb MPG`)^(-1)
epaModel_NEW <- lm(transformedCmbMPG ~ Displ + `Veh Class`)
plot(resid(epaModel_NEW) ~ fitted(epaModel_NEW))
```




```
qqPlot(resid(epaModel_NEW))
```




```
[1] 1361 1362
```

This improves the unequal variance in the errors further, but still does not help with the non-normality.

We could attempt further transformations (and I would encourage you to try a few – try some numbers other than -1 in the code above). However after some trial and error you'll likely find that we cannot seem to meet the normality assumption.

This likely indicates there is something missing from the model, or that regression is simply not the correct technique to use here. Remember, this is your first course in this program and although all we've seen so far

is regression, there is a lot more out there which you will see as you continue through the program.

For now, if you come across this in a problem, simply be careful to note the problem and indicate you have done your best to fix it. Then when using the model we will need to be aware that its accuracy may be compromised.

In this case it appears the best model we can make is the y^{-1} model. That satisfies the equal variances requirement the best, however we note that it violates the assumption of normally distributed errors.