Simple Linear Regression: Conditions and Transformations

Sleuth3 Chapter 8

Example 1: Exercise 8.17 in Sleuth3

Quote from book:

> In a study of the effectiveness of biological control fo the exotic weed tansy ragwort, researchers manipulated the exposure to the ragwort flea beetle on 15 plots that had been planted with a high density of ragword. Havesting the plots the next season, they measured the average dry mass of ragword remaining (gams/plant) and the flea beetle load (beetles/gram of ragword dry mass) to see if the ragwort plants in plots with high flea beetle loads were smaller as a result of herbivory by the beetles. (Data from P. McEvoy and C. Cox, "Successful Biological Control of Ragwort, *Senecio jacobaea, by Introduced Insects in Oregon,",* Ecological Applications* 1 (4) (1991): 430-42.)

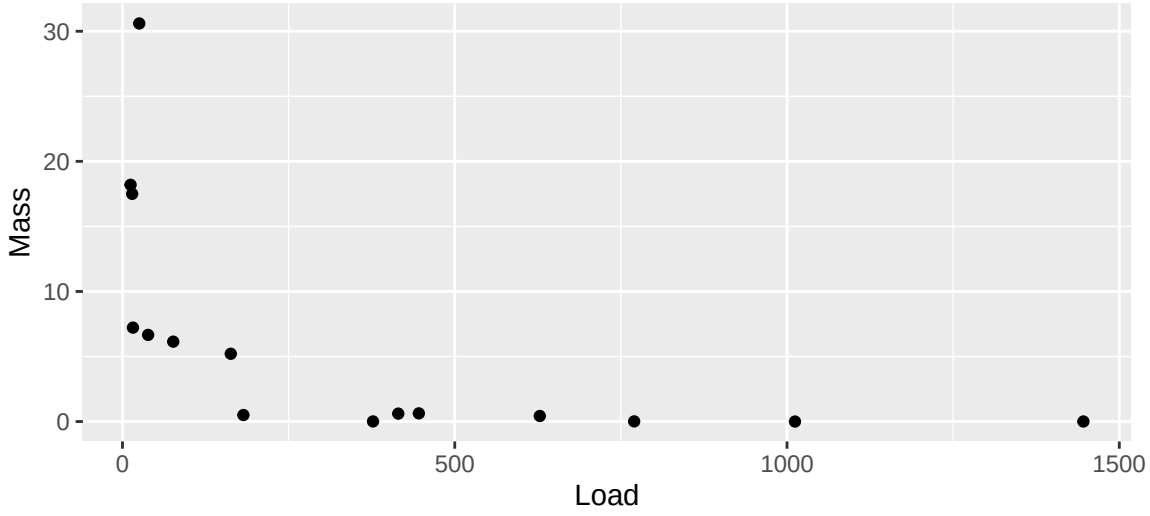
Here is the data:

```
## # A tibble: 6 x 2
##    Load  Mass
##   <dbl> <dbl>
## 1  12.2 18.2
## 2  14.6 17.5
## 3  15.8  7.22
## 4  25.3 30.6
## 5  38.6  6.66
## 6  76.4  6.14
```

Our explanatory variable is `Load`, and the response is `Mass`.

1. Make a suitable plot of the data.

```
ggplot(data = pest_control, mapping = aes(x = Load, y = Mass)) +
  geom_point()
```



2. Through trial and error, find a suitable transformation of the data so that the linear regression conditions are satisfied as well as possible. (Let's assume the plots were in different areas so that they can be regarded as independent.)

For the purpose of this lab, I'll give you a little more instruction than I would on a homework assignment:

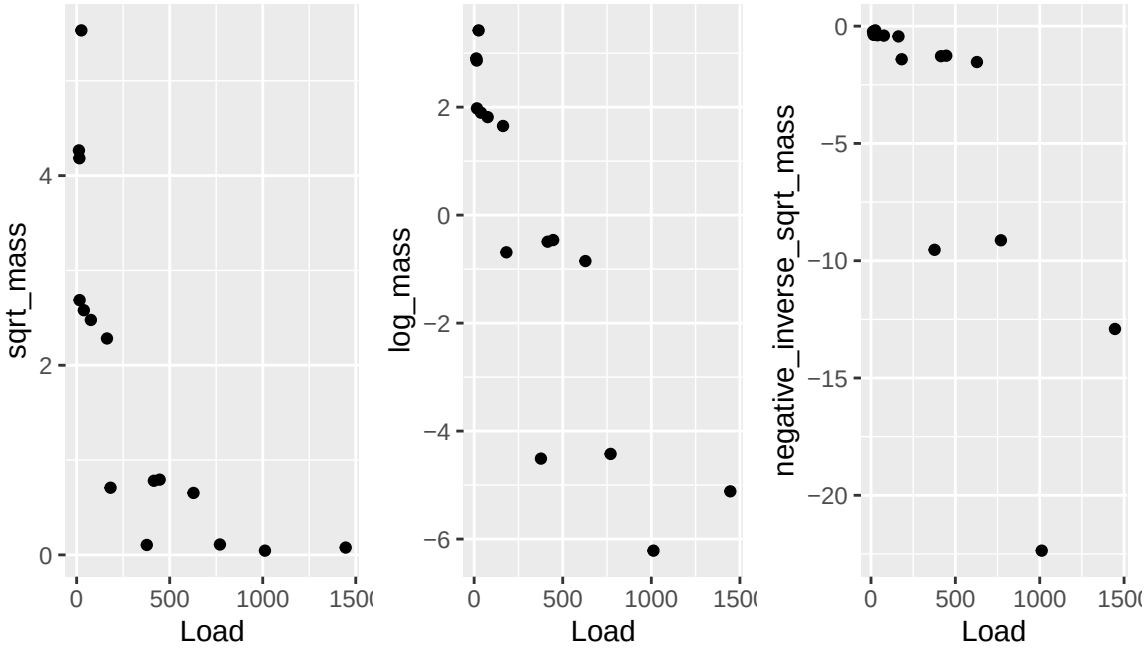
- You should start by transforming the response variable, aiming to fix the problem with unequal standard deviations of the residuals. I ended up with a transformation of mass to the power of 0.25 – your goal for today is to understand why and see how you could arrive at that transformation yourself.

- Then, once you have pretty constant standard deviation of the residuals, if necessary you can try transformations of the explanatory variables until you have a relationship that looks closer to linear. I chose a log transformation for the load variable. Again, your goal is to understand why and how you could identify that transformation yourself through experimentation.
- For each transformation you try, you should make a scatter plot of the transformed variables, and a scatter plot of the transformed explanatory variable vs. the residuals from a simple linear regression model fit to the transformed variables.

The first thing I see is that the standard deviation of the response variable is not equal across the range of values for the explanatory variable. This suggests that we should start by considering transformations of the response variable. Since the response variable is skewed right (many small values of Mass, a few large outlying values), I will move down the ladder of powers.

I'm showing the next three steps down on the ladder here:

```
pest_control <- pest_control %>%
  mutate(
    sqrt_mass = sqrt(Mass),
    log_mass = log(Mass),
    negative_inverse_sqrt_mass = -1/sqrt(Mass)
  )
p_sqrt <- ggplot(data = pest_control, mapping = aes(x = Load, y = sqrt_mass)) +
  geom_point()
p_log <- ggplot(data = pest_control, mapping = aes(x = Load, y = log_mass)) +
  geom_point()
p_neg_inv_sqrt <- ggplot(data = pest_control, mapping = aes(x = Load, y = negative_inverse_sqrt_mass)) +
  geom_point()
grid.arrange(
  p_sqrt,
  p_log,
  p_neg_inv_sqrt,
  nrow = 1
)
```

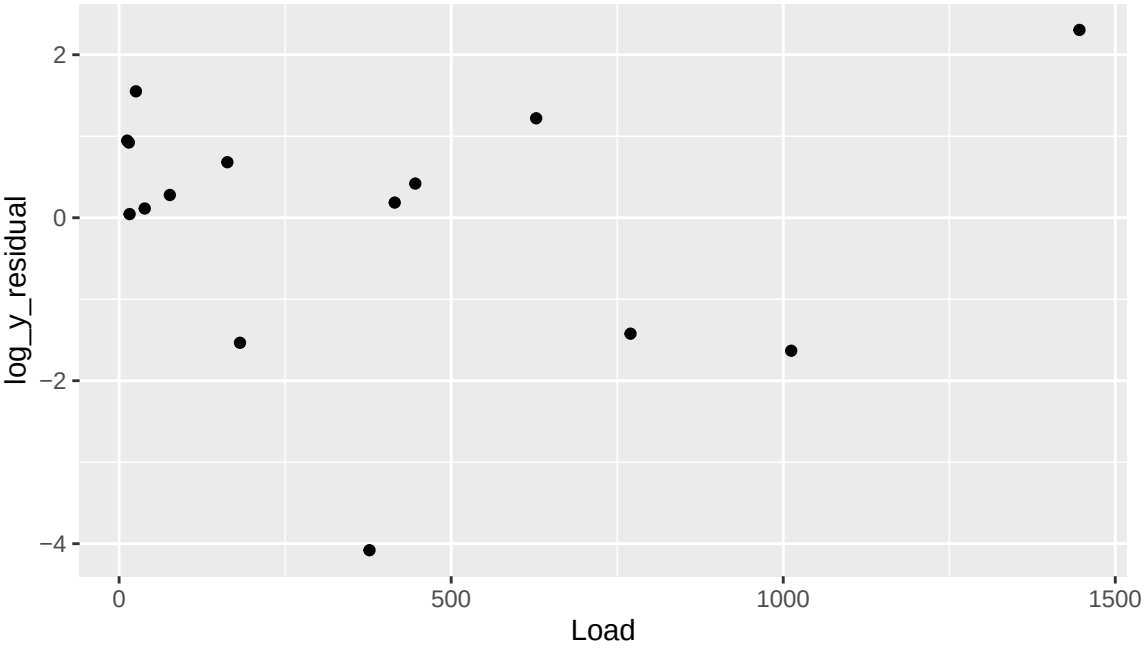


In these plots:

- The square root transformation doesn't seem to have done enough. There is still a larger standard deviation for small values of Load than for large values of Load
- The log transformation looks better, though maybe not perfect.
- The $-1/\sqrt{y}$ transformation went too far: the standard deviation is now too small on the left and too large on the right

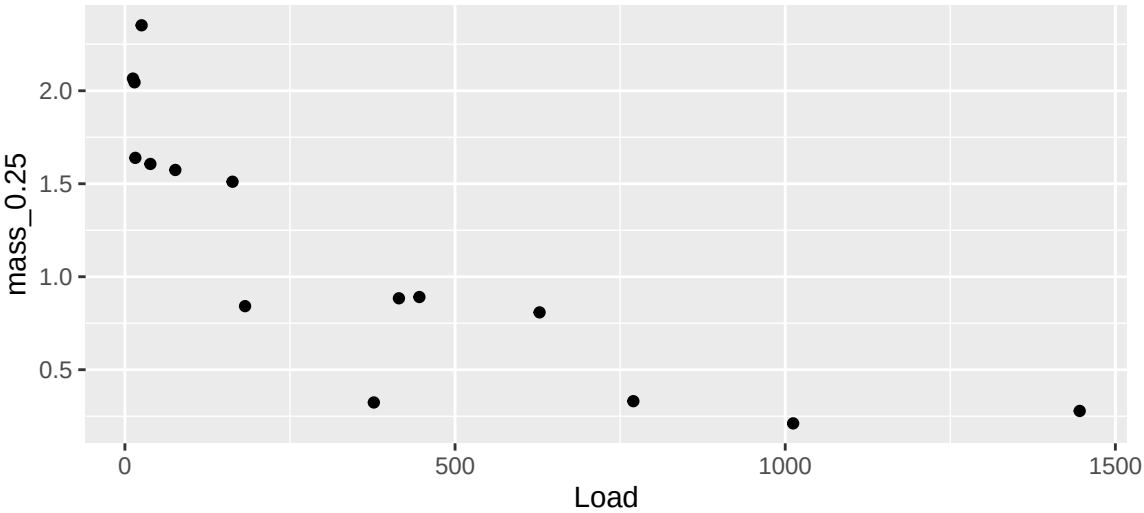
Let's look at that log tranformation more closely:

```
log_y_fit <- lm(log_mass ~ Load, data = pest_control)
pest_control <- pest_control %>%
  mutate(
    log_y_residual = residuals(log_y_fit)
  )
ggplot(data = pest_control, mapping = aes(x = Load, y = log_y_residual)) +
  geom_point()
```



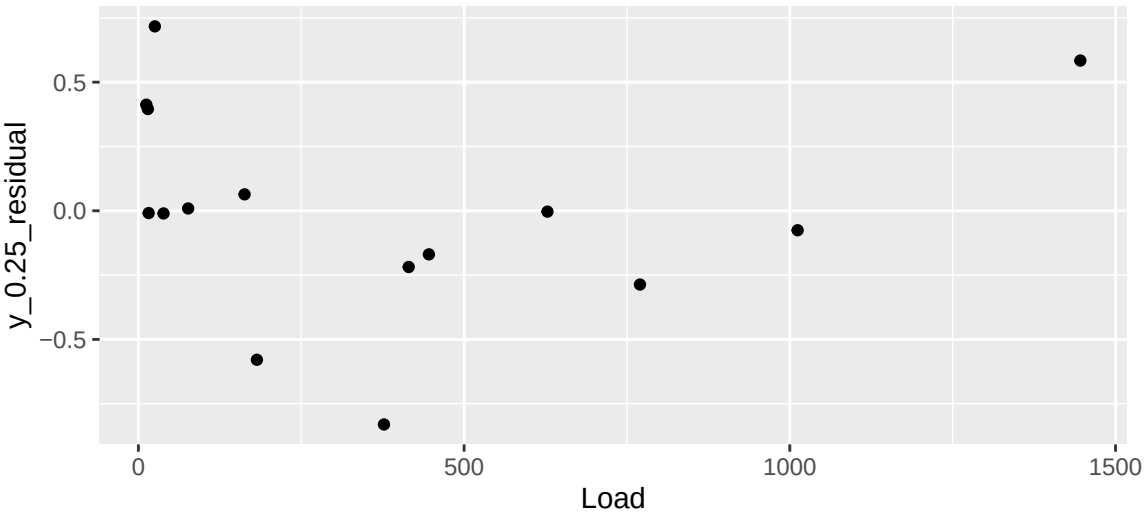
I see evidence of non-linearity here, as well as a smaller standard deviation for small Loads then for large Loads. Let's continue trying to fix the non-equal standard deviations. Since the square root transformation had too-large residuals on the left and the log transformation has too-small residuals on the left, let's try something inbetween those two: $y^{-0.25}$.

```
pest_control <- pest_control %>%
  mutate(
    mass_0.25 = Mass^0.25
  )
ggplot(data = pest_control, mapping = aes(x = Load, y = mass_0.25)) +
  geom_point()
```




```
y_0.25_fit <- lm(mass_0.25 ~ Load, data = pest_control)
pest_control <- pest_control %>%
  mutate(
    y_0.25_residual = residuals(y_0.25_fit)
```

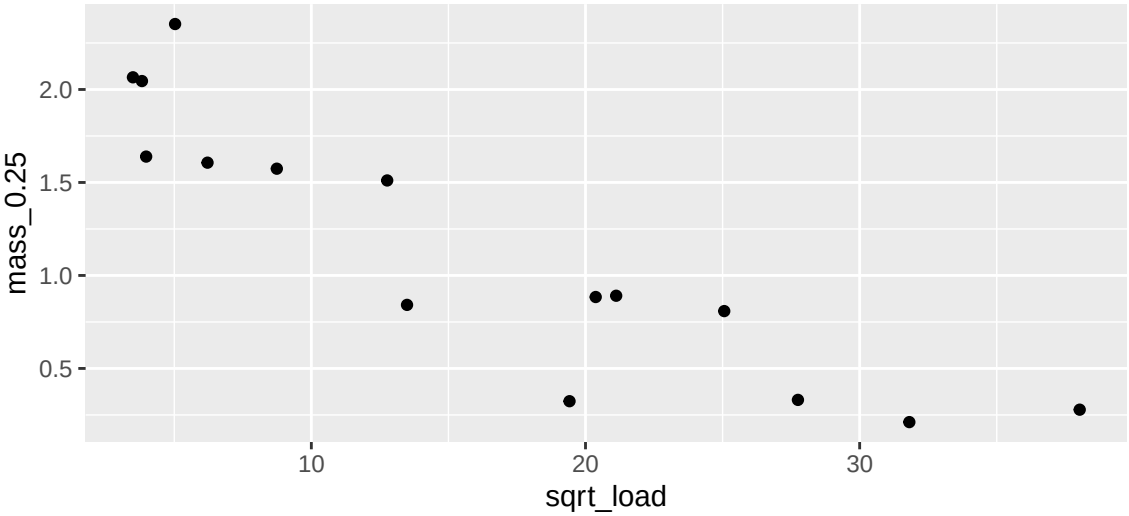
```
  )
ggplot(data = pest_control, mapping = aes(x = Load, y = y_0.25_residual)) +
  geom_point()
```



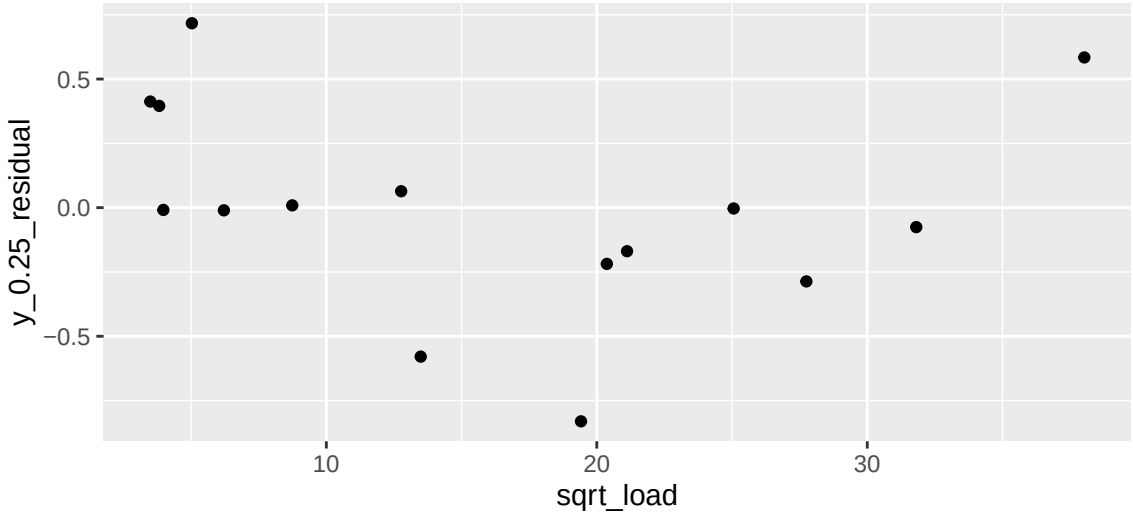
We are now in a place where the residual standard deviation is roughly equal across the range of values for load (at least, in the region where there is enough data to assess this).

We still have a problem with non-linearity. We don't want to change the response variable any more though. We note that the Load variable is skewed right (many small values for Load, and a few large values). Let's try moving down the ladder of powers to pull in the outlying values.

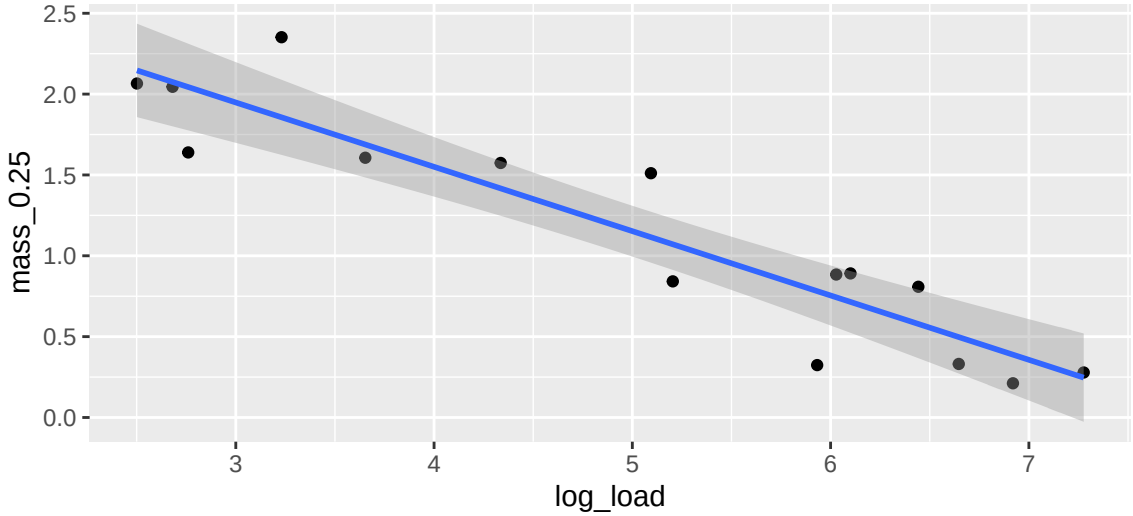
```
pest_control <- pest_control %>%
  mutate(
    sqrt_load = Load^0.5,
    log_load = log(Load)
  )
ggplot(data = pest_control, mapping = aes(x = sqrt_load, y = mass_0.25)) +
  geom_point()
```



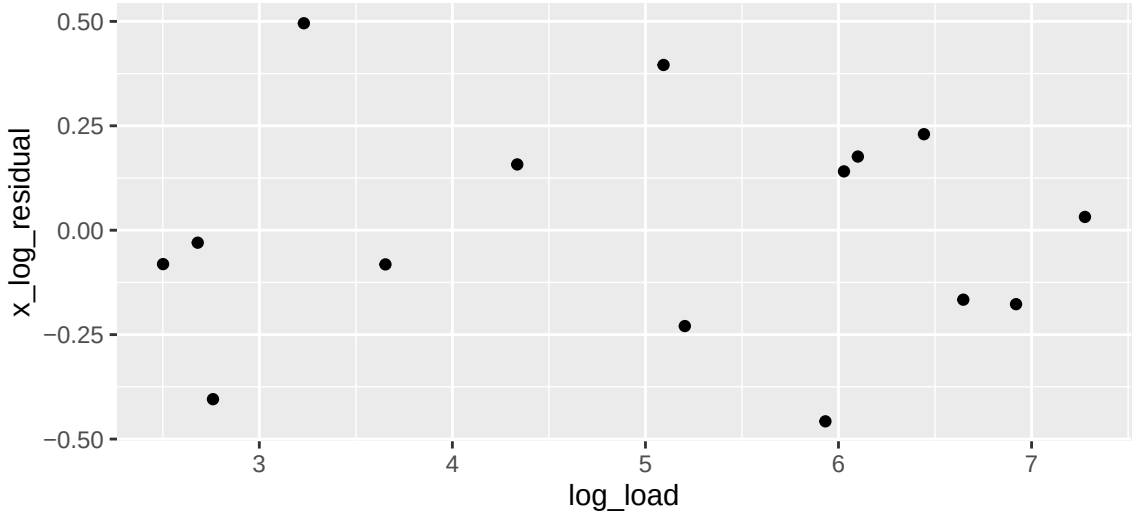

```
x_0.5_fit <- lm(mass_0.25 ~ sqrt_load, data = pest_control)
pest_control <- pest_control %>%
  mutate(
    x_0.5_residual = residuals(x_0.5_fit)
  )
ggplot(data = pest_control, mapping = aes(x = sqrt_load, y = y_0.25_residual)) +
  geom_point()
```




```
ggplot(data = pest_control, mapping = aes(x = log_load, y = mass_0.25)) +
  geom_point() +
  geom_smooth(method = "lm")
```

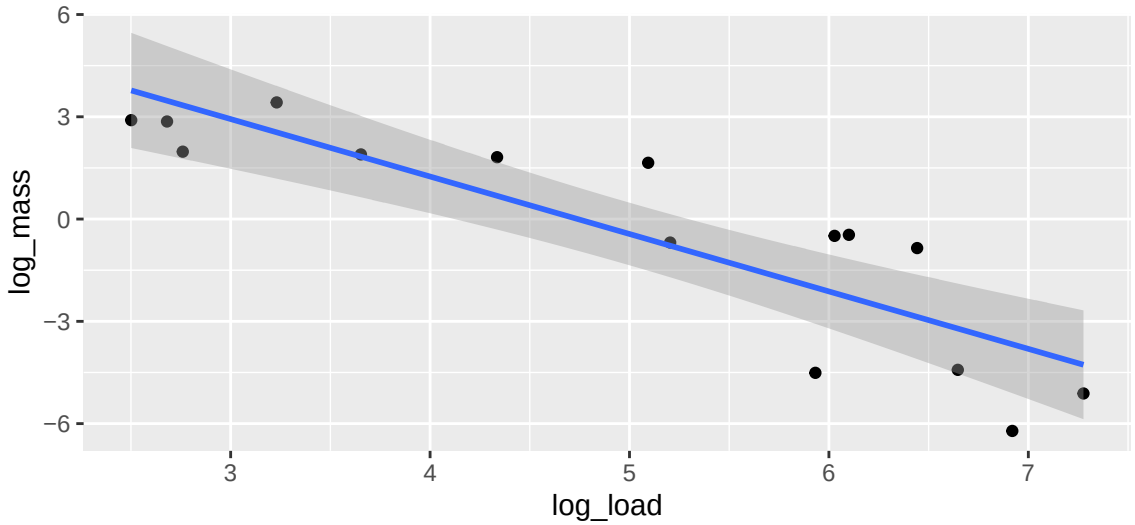



```
x_log_fit <- lm(mass_0.25 ~ log_load, data = pest_control)
pest_control <- pest_control %>%
  mutate(
    x_log_residual = residuals(x_log_fit)
  )
ggplot(data = pest_control, mapping = aes(x = log_load, y = x_log_residual)) +
  geom_point()
```



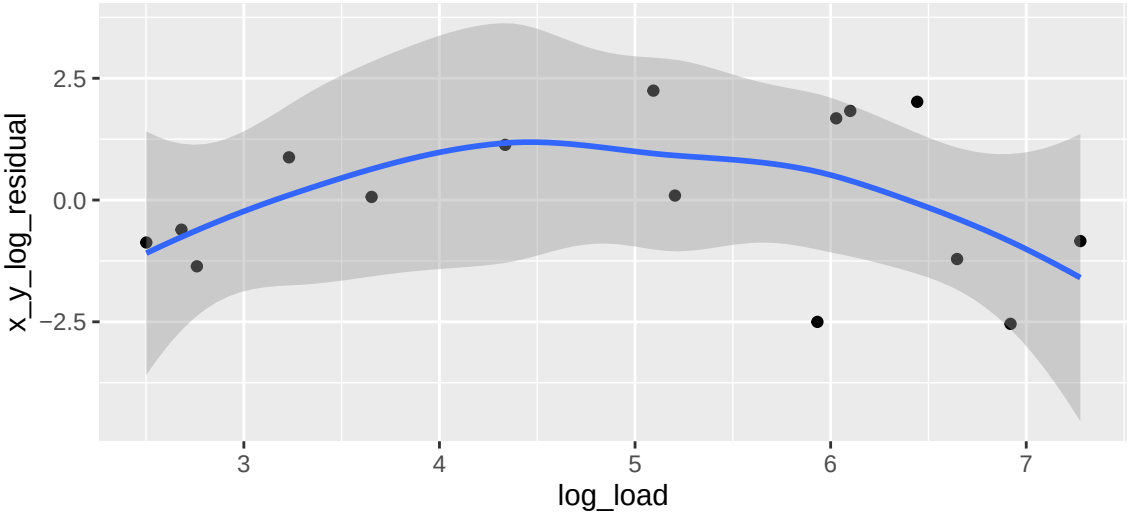
It would really be more convenient if we could use a log transformation for both… can we?

```
ggplot(data = pest_control, mapping = aes(x = log_load, y = log_mass)) +
  geom_point() +
  geom_smooth(method = "lm")
```




```
x_y_log_fit <- lm(log_mass ~ log_load, data = pest_control)
pest_control <- pest_control %>%
  mutate(
    x_y_log_residual = residuals(x_y_log_fit)
  )
ggplot(data = pest_control, mapping = aes(x = log_load, y = x_y_log_residual)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
pest_control <- pest_control %>%
  mutate(
    group = ifelse(log_load < 5, "small load", "large load")
  )
pest_control %>%
  group_by(group) %>%
  summarize(sd = sd(x_y_log_residual))
```

```
## # A tibble: 2 x 2
##   group         sd
##   <chr>      <dbl>
## 1 large load 1.94
## 2 small load 0.994
```

The log transformation is sortof ok, but just doesn't seem as good as the transformation to a power of 0.25.

3. Conduct a test of the claim that there is no association between the beetles load and the mean dry mass of ragweed harvested (after transformation).

```
x_log_fit <- lm(mass_0.25 ~ log_load, data = pest_control)
summary(x_log_fit)
```

```
##
## Call:
## lm(formula = mass_0.25 ~ log_load, data = pest_control)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.45759 -0.17168 -0.02985  0.16693  0.49557
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.14198    0.23669   13.28 6.17e-09 ***
## log_load    -0.39792    0.04517   -8.81 7.66e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2816 on 13 degrees of freedom
## Multiple R-squared:  0.8565, Adjusted R-squared:  0.8455
## F-statistic: 77.62 on 1 and 13 DF,  p-value: 7.659e-07
```

$H_0 : \beta_1 = 0$: There is no linear association between log-transformed beetles load and ragweed mass to the power of 1/4.

$H_A : \beta_1 \neq 0$ There is a linear association between log-transformed beetles load and ragweed mass to the power of 1/4.

The p-value for this test is 7.66e-07. The data provide extremely strong evidence against the null hypothesis of no linear association between log-transformed beetles load and ragweed mass to the power of 1/4.

4. Find an estimate of the median ragweed mass when the beetle load is 50, and when it is 250. Additionally, report a confidence interval for the median ragweed mass when the beetle load is 250.

First: apply the transformation to the explanatory variable, save in a data frame

```
predict_data <- data.frame(
  Load = c(50, 250)
) %>%
  mutate(
    log_load = log(Load)
  )
predict_data
```

```
##   Load log_load
## 1   50 3.912023
## 2  250 5.521461
```

Second: generate a prediction/estimate for the mean of $Mass^{0.25}$

```
predict(x_log_fit, newdata = predict_data, level = 0.95, interval = "confidence")
```

```
##         fit       lwr      upr
## 1 1.5853238 1.3964518 1.774196
## 2 0.9449032 0.7794105 1.110396
```

Third: Undo the transformation.

Point estimates:

```
1.5853238^4
```

```
## [1] 6.316433
```

```
0.9449032^4
```

```
## [1] 0.7971669
```

Confidence interval for median when Load = 250:

```
0.7794105^4
```

```
## [1] 0.3690328
```

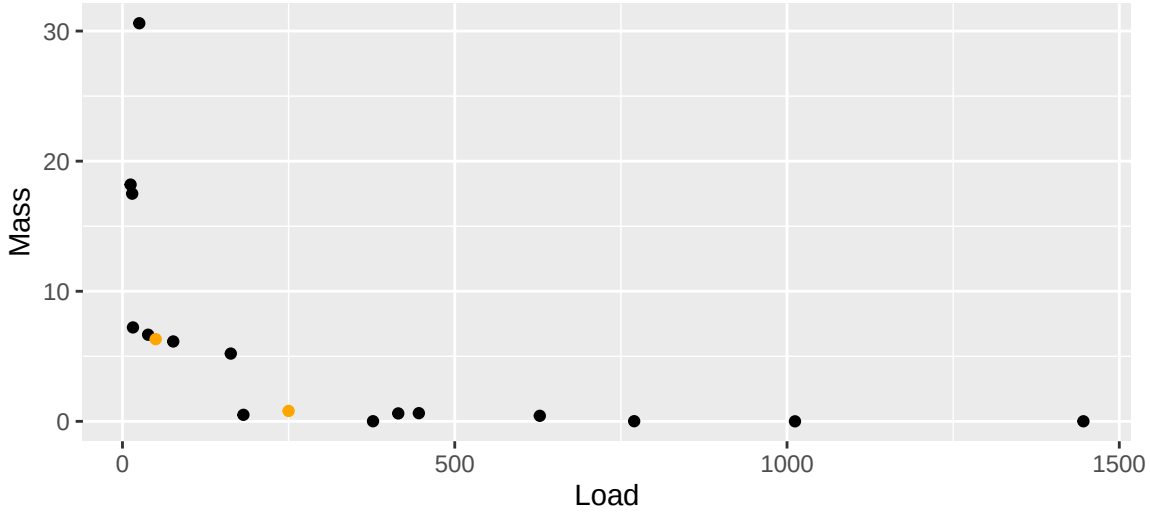
```
1.110396^4
```

```
## [1] 1.520238
```

We estimate that the median ragweed mass in fields where the beetles load is 250 beetles/gram of ragword dry mass is about 0.797 grams/plant. We are 95% confident that this median is between about 0.369 and 1.520.

Here is a plot:

```
predict_data <- predict_data %>%
  mutate(
    Mass = c(6.32, 0.80)
  )
ggplot(data = pest_control, mapping = aes(x = Load, y = Mass)) +
  geom_point() +
  geom_point(data = predict_data, color = "orange")
```



Example 2: Adapted from Exercise 8.24 in Sleuth3

Quote from the book:

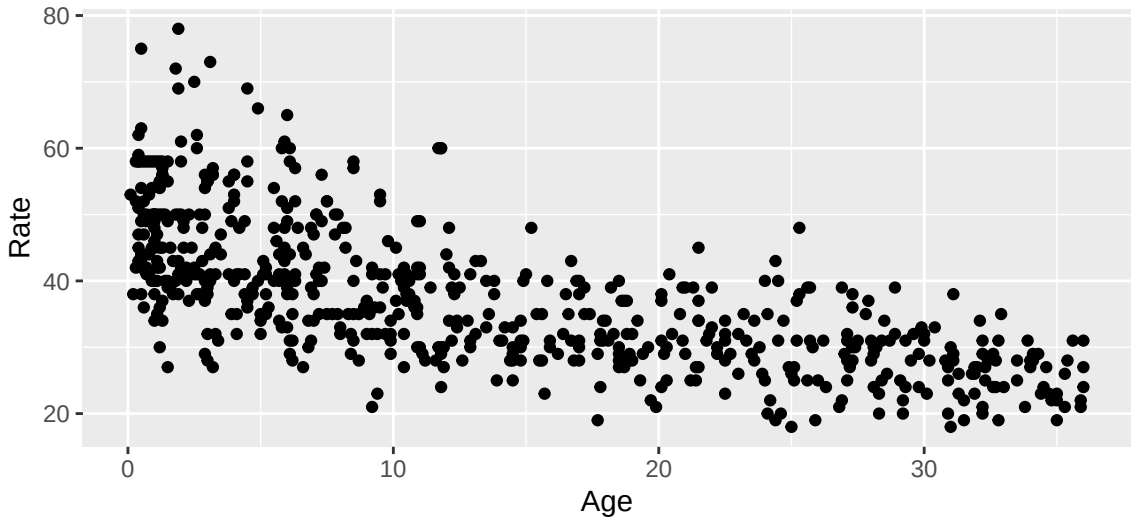
> A high respiratory rate is a potential diagnostic indicator of respiratory infection in children. To judge whether a respiratory rate is truly "high", however, a physician must have a clear picture of the distribution of normal respiratory rates. To this end, Italian researchers measured the respiratory rates of 618 children between the ages of 15 days and 3 years. Analyze the data and provide a statistical summary. The following R code reads the data in.

```
## # A tibble: 6 x 2
##     Age  Rate
##   <dbl> <dbl>
## 1   0.1    53
## 2   0.2    38
## 3   0.3    58
## 4   0.3    52
## 5   0.3    42
## 6   0.4    62
```

Our explanatory variable is `Age` (in months), and the response is `Rate` (breaths per minute).

1. Make a suitable plot of the data.

```
ggplot(data = respiration, mapping = aes(x = Age, y = Rate)) +
  geom_point()
```

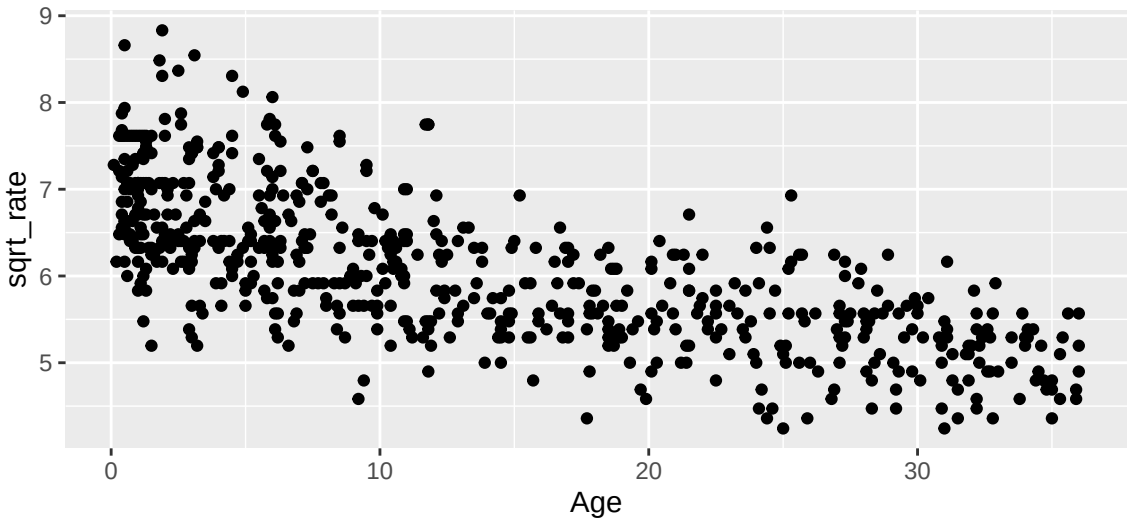


2. Through trial and error, find a suitable transformation of the data so that the linear regression conditions are satisfied as well as possible. (Let's assume the measurements for different children in the sample can be regarded as independent.)

I selected transformations of $-1/\sqrt{Rate}$ for the response and $Age^{0.75}$ for the explanatory variable. Again, your goal is to understand the process for getting to those (or similar) transformations.

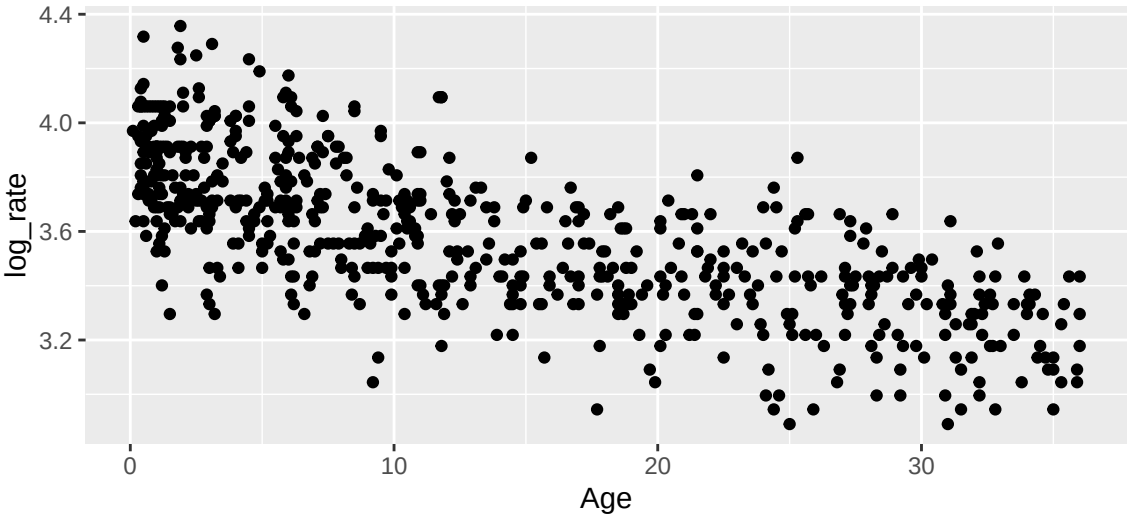
In the initial plot, we see a larger standard deviation for the response for small ages, and a smaller standard deviation for the response for larger ages. This suggests that we try a transformation of the response variable. That variable is skewed to the right, so we should move down the ladder. I will start with a square root transformation:

```
respiration <- respiration %>%
  mutate(
    sqrt_rate = sqrt(Rate)
  )
ggplot(data = respiration, mapping = aes(x = Age, y = sqrt_rate)) +
  geom_point()
```



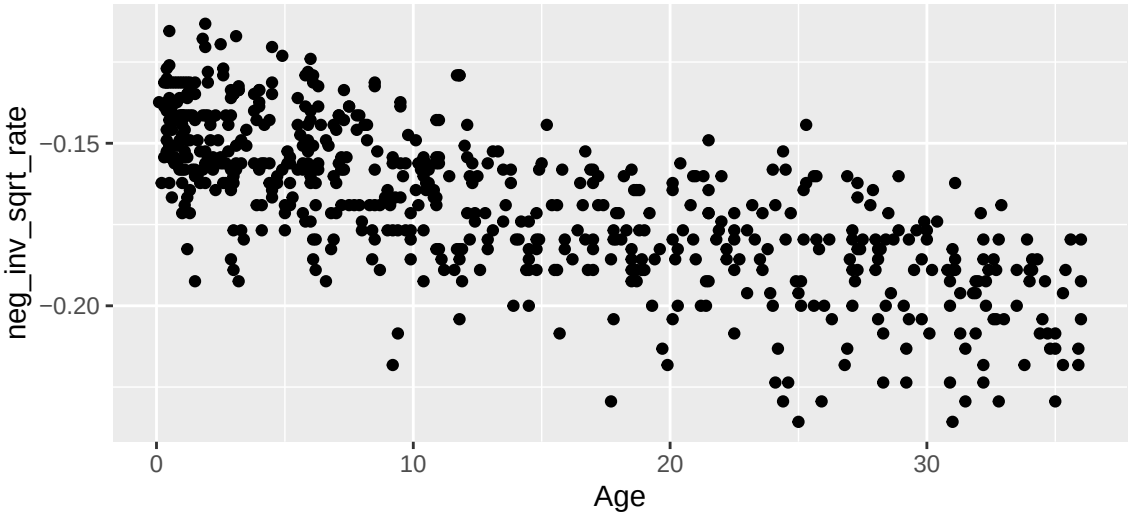
This is better, but not good enough. Let's try a log transformation:

```
respiration <- respiration %>%
  mutate(
    log_rate = log(Rate)
  )
ggplot(data = respiration, mapping = aes(x = Age, y = log_rate)) +
  geom_point()
```



This is still not quite good enough. One more step down?

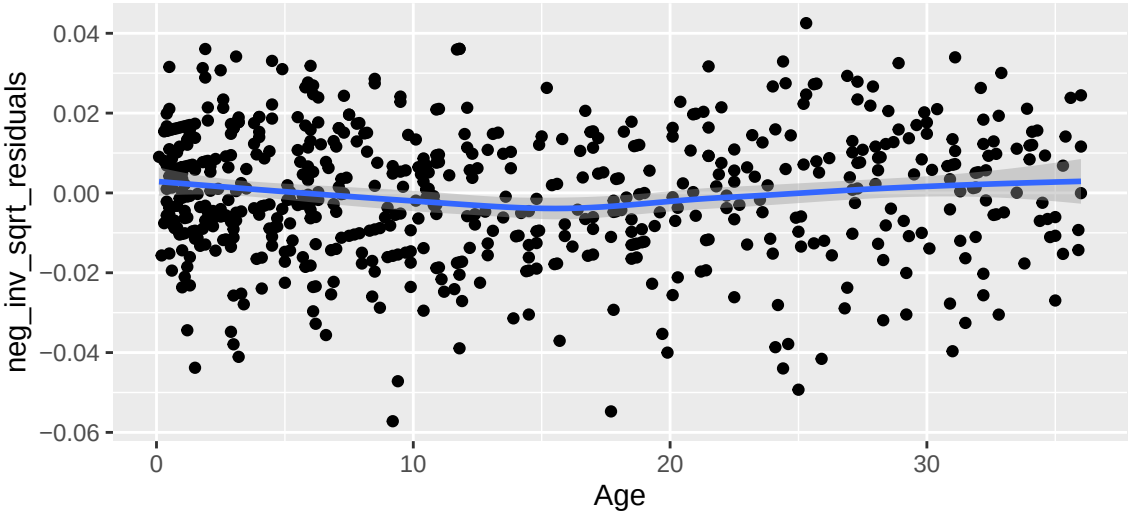
```
respiration <- respiration %>%
  mutate(
    neg_inv_sqrt_rate = -1/sqrt(Rate)
  )
ggplot(data = respiration, mapping = aes(x = Age, y = neg_inv_sqrt_rate)) +
  geom_point()
```



Let's fit a line to the data with this transformation and examine the residuals to see how we're doing.

```
lm_fit <- lm(neg_inv_sqrt_rate ~ Age, data = respiration)
respiration <- respiration %>%
  mutate(
    neg_inv_sqrt_residuals = residuals(lm_fit)
  )
ggplot(data = respiration, mapping = aes(x = Age, y = neg_inv_sqrt_residuals)) +
  geom_point() +
  geom_smooth()
```

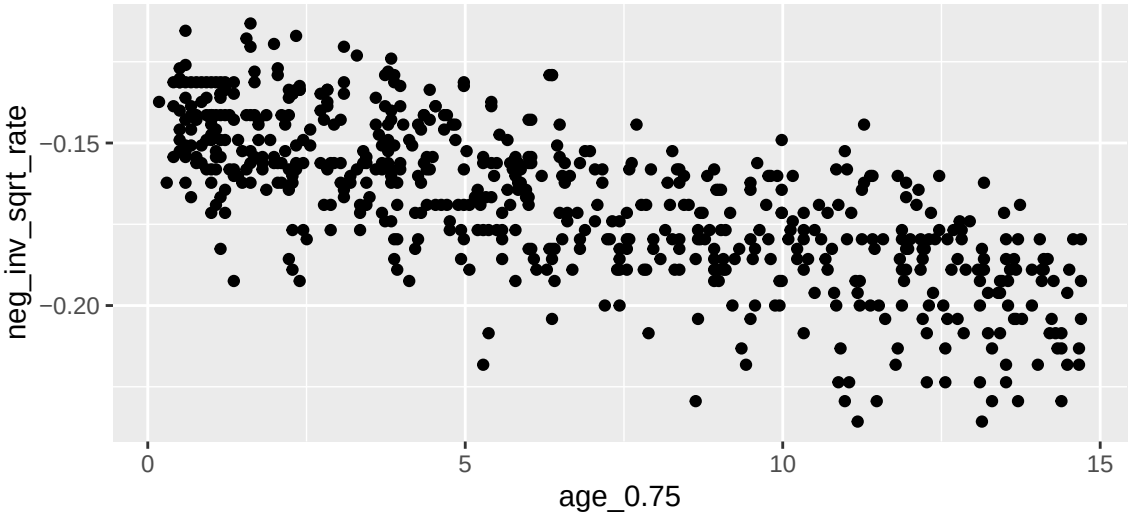
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



This residuals plot shows a consistent standard deviation of the residuals across all values of Age. There is a very slight indication of curvature. The relationship is close enough to linear at this point that the resulting model would probably be good enough for most purposes. If we really wanted to we could try adusting Age as well.

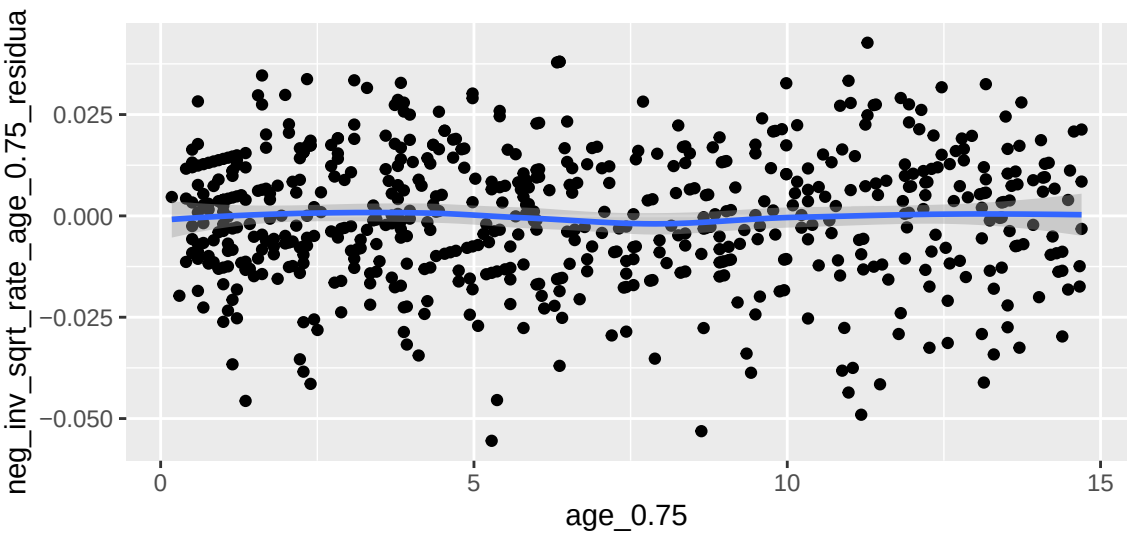
```
respiration <- respiration %>%
  mutate(
    age_0.75 = Age^0.75
  )
```

```
ggplot(data = respiration, mapping = aes(x = age_0.75, y = neg_inv_sqrt_rate)) +
  geom_point()
```




```
lm_fit <- lm(neg_inv_sqrt_rate ~ age_0.75, data = respiration)
respiration <- respiration %>%
  mutate(
    neg_inv_sqrt_rate_age_0.75_residuals = residuals(lm_fit)
  )
ggplot(data = respiration, mapping = aes(x = age_0.75, y = neg_inv_sqrt_rate_age_0.75_residuals)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



This model is ever so slightly better than the model with Age as the explanatory variable. Enough better that it's worth the effort of dealing with the transformation of the explanatory variable? Unclear.

3. Obtain 95% prediction intervals for the respiratory rates *on the original data scale* for children of ages 5 months, 10 months, 20 months, and 30 months. Add a display of your prediction intervals to a scatter plot of the original data (you can copy/paste your plot code from part 1 and add a layer for the prediction intervals).

I'm going to work with the model where I had transformed age just to demonstrate the ideas. If we hadn't transformed the explanatory variable age, everything would be the same but we would not need to transform age in the first step.

The first thing I will do is set up a data frame with the ages at which we want to make predictions, and apply the selected transformation to those ages.

```
predict_data <- data.frame(
  Age = c(5, 10, 20, 30)
) %>%
  mutate(
    age_0.75 = Age^0.75
  )
```

Next, we will use our final model, with $Age^{0.75}$ as the explanatory variable and $-1/\sqrt{Rate}$ as the response, to generate predictions. These will be predictions for $-1/\sqrt{Rate}$

```
predict(lm_fit, newdata = predict_data, interval = "prediction")
```

```
##          fit        lwr        upr
## 1 -0.1548627 -0.1867942 -0.1229313
## 2 -0.1641093 -0.1960276 -0.1321909
## 3 -0.1796600 -0.2115893 -0.1477306
## 4 -0.1932930 -0.2252660 -0.1613200
```

We now need to undo the transformation of the response. If $Y = -1/\sqrt{Rate}$, then $\sqrt{Rate} = -1/Y$, or $Rate = 1/Y^2$. So to get back to rate, we need to take the predicted values above, square them, and then take the reciprocal of that. We can do this by converting the results above to a data frame and then using mutate on the columns of that data frame.

```
predictions <- predict(lm_fit, newdata = predict_data, interval = "prediction") %>%
  as.data.frame() %>%
  mutate(
    fit_original = 1/fit^2,
    lwr_original = 1/lwr^2,
    upr_original = 1/upr^2,
    Age = c(5, 10, 20, 30)
  )
predictions
```

```
##          fit        lwr        upr fit_original lwr_original upr_original
## 1 -0.1548627 -0.1867942 -0.1229313     41.69713     28.65982     66.17212
## 2 -0.1641093 -0.1960276 -0.1321909     37.13076     26.02349     57.22647
## 3 -0.1796600 -0.2115893 -0.1477306     30.98113     22.33636     45.82041
## 4 -0.1932930 -0.2252660 -0.1613200     26.76503     19.70646     38.42586
##   Age
## 1   5
## 2  10
## 3  20
## 4  30
```

We are 95% confident that a child of age 5 months will have a respiratory rate between 28.66 and 66.17 breaths per minute.

We are 95% confident that a child of age 10 months will have a respiratory rate between 26.02 and 57.23 breaths per minute.

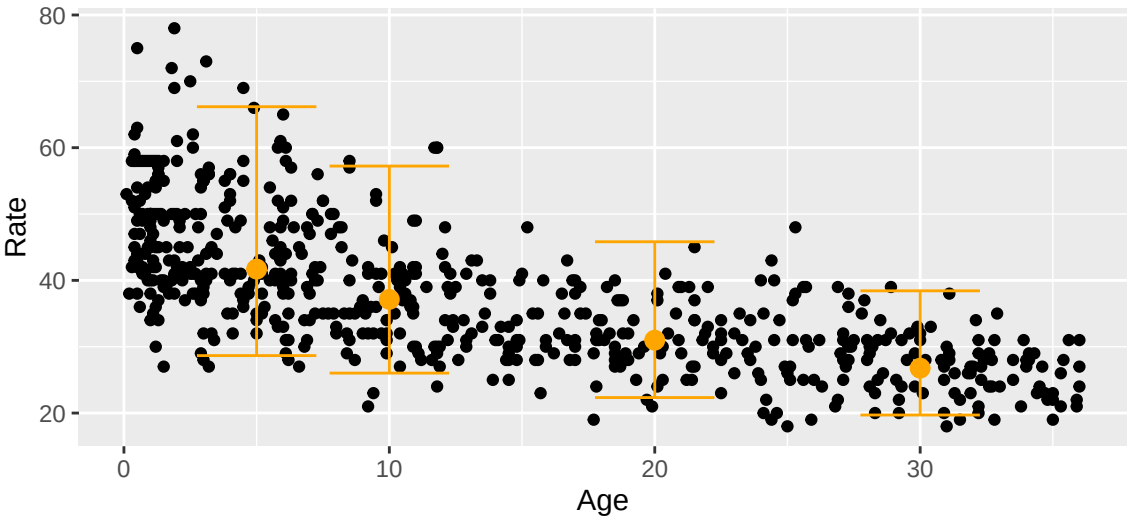
We are 95% confident that a child of age 20 months will have a respiratory rate between 22.34 and 45.82 breaths per minute.

We are 95% confident that a child of age 30 months will have a respiratory rate between 19.71 and 38.43 breaths per minute.

(These are four separate prediction intervals.)

Here is a plot of the original data with these intervals overlaid on top.

```
ggplot() +
  geom_point(data = respiration, mapping = aes(x = Age, y = Rate)) +
  geom_point(data = predictions, mapping = aes(x = Age, y = fit_original), color = "orange", size = 3) +
  geom_errorbar(data = predictions, mapping = aes(x = Age, ymin = lwr_original, ymax = upr_original), color =
```



Each of the prediction intervals does appear to contain most, but not all, of the observed values around the corresponding age. Note that the intervals at smaller ages are wider than the intervals at larger ages; this corresponds to the fact that the data have a higher standard deviation for small ages than for large ages. Also note that the intervals are asymmetric; they are longer on the top than on the bottom. This also matches a feature of the data, that the rates are skewed right for each age.