Hacky Hour 20231114 - Linear Regression

Goal of Today: Linear Regression

Hi y'all. Today at Hacky Hour, we'll learn a bit about running linear regression in R. For today's data, we're using a subset of data from TidyTuesday last year. You can find the entire data set here from Tanya Shapiro. Through this lesson, we will learn:

1. How to make and interpret a linear model in R
2. How to assess the *quality* of your linear model
3. When transformations may be necessary
4. How to include a linear model in a ggplot

Reading in Our Data

We can read in our data from the data folder. We'll use the `read_csv()` function from `readr`; I'm going to load the entire `tidyverse` package because I often use functions from multiple packages within it. Also I set a new default theme in ggplot because the default one is just ugly

```
library(tidyverse) # Load the tidyverse packages

theme_set(theme_classic()) # Set our ggplot theme to apply to all plots



horror_data <- read_csv("data/horror_movies.csv") # Read in our data
```

```
glimpse(horror_data)
```

```
Rows: 526
Columns: 7
$ title        <chr> "The Black Phone", "Nope", "X", "Halloween Kills", "Screa~
$ release_date <date> 2022-06-22, 2022-07-20, 2022-03-17, 2021-10-14, 2022-01-~
$ revenue      <dbl> 161000000, 170800000, 14257609, 131647155, 140000000, 779~
$ budget       <dbl> 1.88e+07, 6.80e+07, 1.00e+07, 2.00e+07, 2.40e+07, 2.00e+0~
$ runtime      <dbl> 103, 130, 106, 105, 114, 123, 100, 111, 107, 170, 99, 103~
$ popularity   <dbl> 1071.398, 733.112, 543.670, 315.027, 291.196, 265.858, 24~
$ vote_count   <dbl> 2736, 1684, 1035, 1941, 1732, 4403, 692, 4689, 1831, 7221~
```

Our data has seven columns, including the title, release date, the revenue it earned, its budget, its run-time, and then a popularity score based on user votes from the website TMDB.

We are interested in one question:

1. How strongly does a movie's budget predict how much revenue it'll earn?

Making a Linear Model in R

To begin, we're going to be bad statisticians, and make a linear model before even LOOKING at our data. I'll make a linear model using the `lm()` function. The simplest linear model includes only one term - the y intercept:

```
single_lm <- lm(revenue ~ 1, data = horror_data)
summary(single_lm)
```

```
Call:
lm(formula = revenue ~ 1, data = horror_data)
```

```
Residuals:
      Min        1Q    Median        3Q       Max
-72138475 -54339076 -27931939  25028060 629694758

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 72147793    3738302    19.3   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 85740000 on 525 degrees of freedom
```

Hmmm, what does this output tell us? If we look at the `Coefficients:` section, we see the term `(Intercept)`, and the value 72147793. I wonder where we got that number?

```
mean(horror_data$revenue)
```

```
[1] 72147793
```

Wow - it's just the mean! If we think of a linear model as giving us the "best guess" for what a value in our response variable will be, it makes sense that if we provide no predictor variables, a model's best guess will be the mean. This isn't very helpful though. This time, let's create a model where revenue is the response variable, and budget is the predictor:

```
rvb_lm <- lm(revenue~budget, data = horror_data)

summary(rvb_lm)
```

```
Call:
lm(formula = revenue ~ budget, data = horror_data)
```

```
Residuals:
       Min          1Q      Median          3Q         Max
-140500531  -37461227  -19753928   16072910  600645308

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.288e+07  4.216e+06   7.799 3.39e-14 ***
budget      1.952e+00  1.376e-01  14.190  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 72940000 on 524 degrees of freedom
Multiple R-squared:  0.2776,    Adjusted R-squared:  0.2762
F-statistic: 201.3 on 1 and 524 DF,  p-value: < 2.2e-16
```

Let's walk through each piece of this output, one by one.

Call:

This stores what code was used to build this model.

Residuals:

This shows the distribution of the residuals, which is how far off the predicted values from our linear model are from the actual data points we observe. *A key assumption of linear regression is that the residuals are normally distributed.* This means that we would expect the residuals would be symmetric around zero, and that the median and mean of the residuals would be the same (zero). Let's test that quickly:

```
mean(rvb_lm$residuals)
```

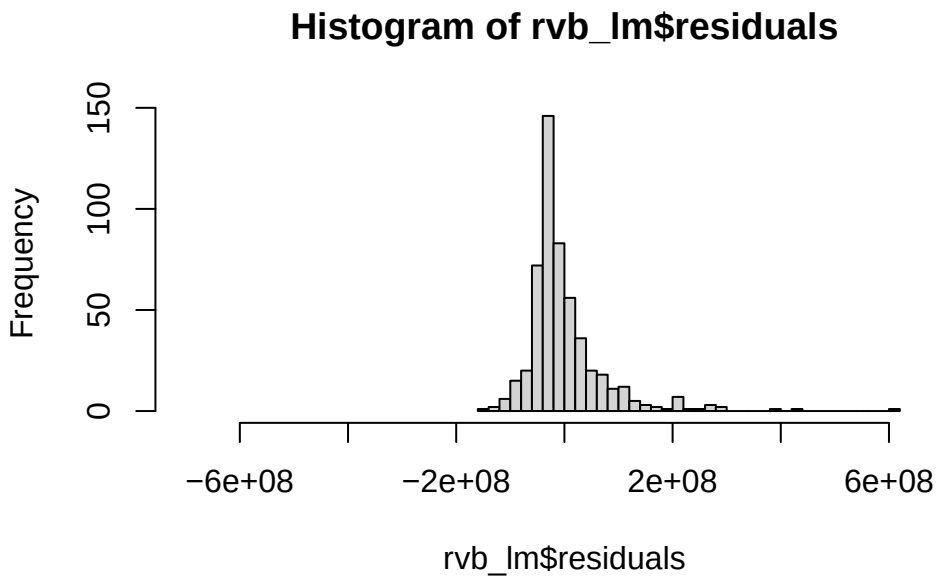
```
[1] 5.286426e-09
```

```r
median(rvb_lm$residuals)
```

```
[1] -19753928
```

Wow - the mean is very close to zero, but the median is not. We can also look at the distribution of residuals to see if they look normal:

```r
hist(rvb_lm$residuals, breaks = 40, xlim = c(-700000000, 700000000)) # Make a histogram
```



As predicted, it looks like our residuals are sitting just left of zero and are right-skewed. This raises some red flags about our model, which we'll discuss later.

Coefficients:

This is the meat of the linear model. If we have a standard linear equation:

$$y = \beta_{\circ} + \beta_1 x$$

$\beta_\circ$ is our y-intercept and β_1 is our slope. R has estimated these terms, which is the `Estimate` column. So our y-intercept ($\beta_\circ$) is in the `(Intercept)` row, and equals $3.288e10^7$. Our slope (β_1) is in the `budget` row and equals 1.952. We can interpret this as: "As budget increases by \$1.00, revenue increases by \$1.95". This is cool - for every dollar spent on a horror movie, we can expect it will earn two dollars back!

We get other information about these terms, including their standard error and the results of a t-test (`t value` and `Pr(>|t|)`), which tells us if these terms appear significantly different from 0 given our data. If there was no relationship between budget and revenue, we would expect the slope to be zero (our null hypothesis). Our low p-value suggests that the slope is significantly different from zero.

`Residual Standard Error:`

This tells us how closely our predicted values match our observed values. We want this number to be as small as possible, and it's in the same units as our response variable (revenue). It's the average distance of each data point from the line of best fit we calculated. So is \$72,940,000 a lot? Let's put that into context with our data:

```
summary(horror_data$revenue)
```

```
   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
   9318  17808717  44215854  72147793  97175852 701842551
```

Our mean movie revenue is \$72,147,793, and the median is \$44,215,854. As such, \$72,940,000 seems like quite a big average distance from the regression line! This is related to the non-normal distribution of residuals as well - the right skew is giving us a very large residual standard error. We'll address this later.

`Multiple R-squared:`

This is everyone's favorite bit of the regression model! In plain terms, the r-squared tells us how much variance in our response variable (revenue) appears to be explained by variance

in our predictor variable (budget). A perfect R-squared is 1, which means every time budget changes, revenue also changes in an exactly linear fashion. An R-squared of 0 means that there is no discernible pattern between budget and revenue - a change in one isn't linked to any change in the other.

What's the difference between "Multiple" and "Adjusted" R squared? The more predictor variables you add to your model, the better the R-squared will be, always. However, these terms might not be adding meaningful information to your model, and you risk "over-fitting", which means you've created a model that perfectly explains your dataset, but might not be generalizable to to the broader population. The adjusted R-squared helps us with that a bit - it calculates the R-squared but penalizes you for more terms, so you can better assess whether adding that new term actually made your model stronger (this is the tip of the iceberg - there are tons of methods for variable selection that we won't touch).

Whether a 0.276 R-squared is "good" will depend strongly on your research question. If you're making a standard curve for a qPCR assay, I've been told nothing less than 0.99 is acceptable. Meanwhile in ecology papers, it's understood that a single variable will only explain a fraction of the variance of the response variable, so 0.276 might reveal a very important relationship.

`F-statistic:`

Finally, the F-statistic. Remember how the t-test was testing whether each term of our model was statistically different from zero? Now with the F-statistic, we're testing if our model as a whole. in this case, the question we're asking is "does there appear to be a relationship between our predictor and response variable - does one change when the other one changes?" If our p-value is low, we can reject the hypothesis that there is no relationship between the variables. Our model (predicting revenue based on budget) is better at predicting revenue than a null model (always just guessing the mean revenue).

Assessing linear model quality

We've done a pretty thorough job of analyzing our output. Here's what we've learned:

1. For every dollar you spent making a movie, we predict it will earn around $2
2. We were able to explain 27.6% of the variance in revenue based on budget
3. Overall our linear model is statistically significant and our budget term is also significantly different from zero; indicating that there is a statistically significant relationship between budget and revenue
4. Our residuals are strongly right-skewed and our residual standard error is high

This last point is a red flag, indicating that our model may have a problem. Maybe the relationship between revenue and budget isn't linear? Maybe revenue and budget aren't normally distributed?

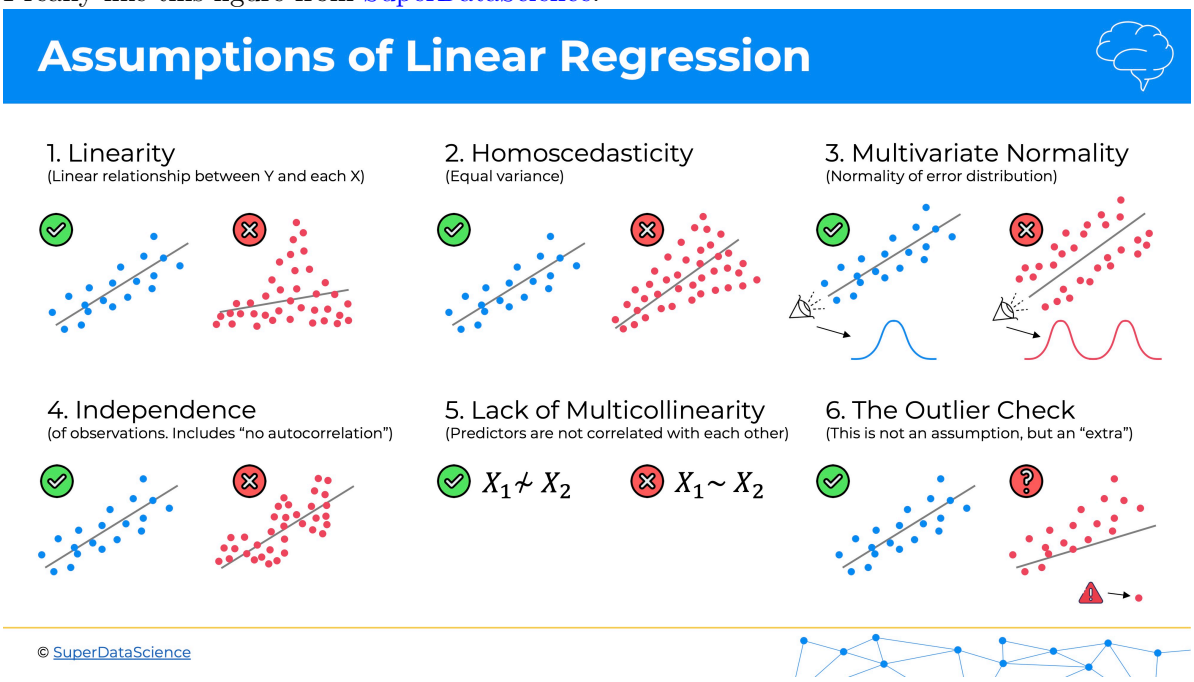
Why does this matter? Like any statistical test, regression is predicated on underlying assumptions:

1. Independence
    1. Our observations need to be independent of each other.
2. Linearity
    1. For a linear model to work, there needs to be a linear relationship between the variables.
3. Homoscedasticity
    1. This is a very fancy word for a relatively simple concept. The variance within our response variable needs to be consistent across our predictor variable.
4. Normality of residuals
    1. Our residuals (the difference between our observed and predicted values) should follow a normal distribution
5. No multicollinearity
    1. This is only relevant for multiple regression, but your predictors shouldn't be correlated with each other

6. No outliers (or at least outliers that came from technical error)

    1. This isn't really an assumption, but like any parametric method, linear regression is vulnerable to outliers

I really like this figure from SuperDataScience:



Let's go through a checklist with our model and see if we're hitting these assumptions:

Independence

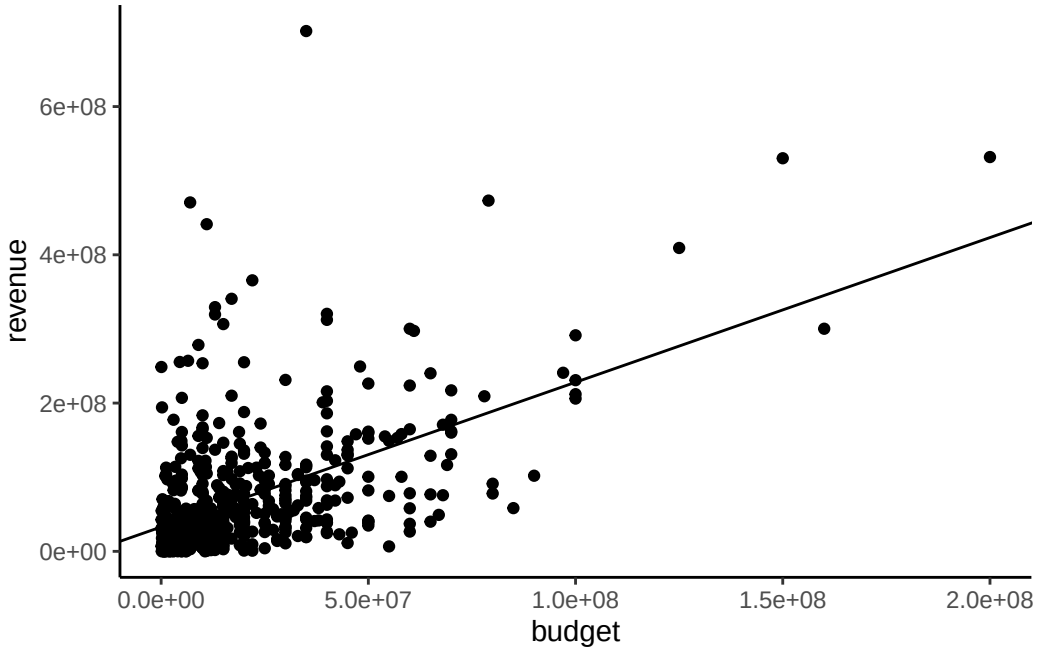
This is really a question of study design. Did our observation of a movie's revenue affect the revenue of movie that came after it? Probably not. We should be okay on this one.

Linearity

Do our variables seem to relate to each other linearly?

Let's plot them to see (hint, this is usually the first thing we should have done). I'm also going to add our regression line by pulling out the coefficients and giving them to geom_abline():

```
horror_data%>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  geom_abline(intercept = rvb_lm$coefficients["(Intercept)"], slope = rvb_lm$coefficients[
```

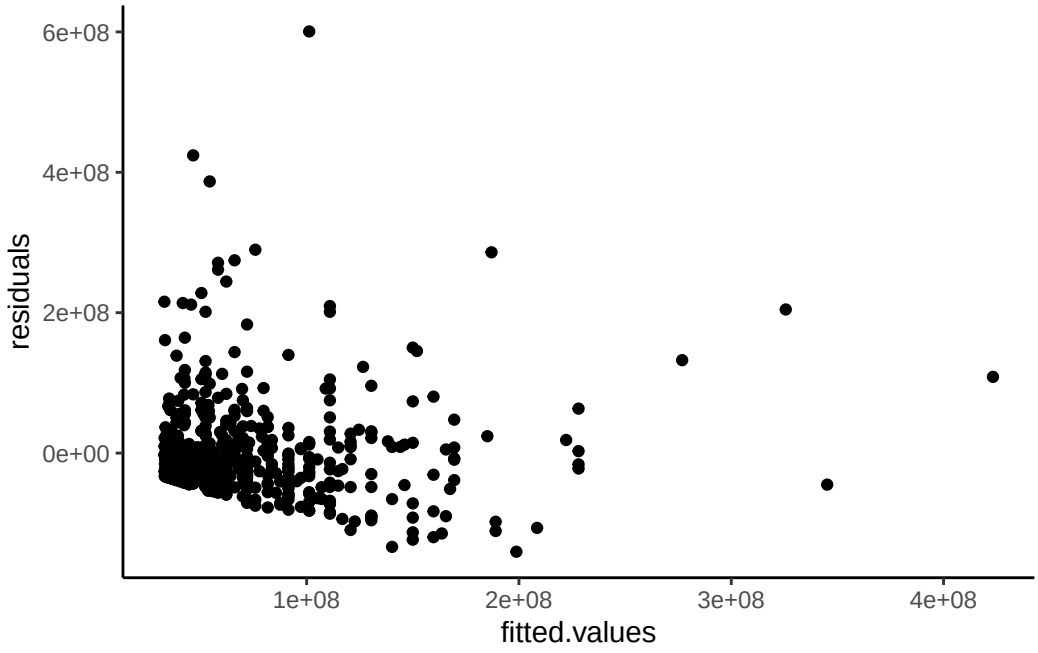


Hmmm, this is tricky. Clearly, there are lots of lower budget, lower revenue movies clustered in the lower left, and then we get a bigger and bigger spread at very high values. This makes it difficult to assess whether there is a linear relationship between the two. Off the bat, however, I don't see any clear evidence of a "curving" relationship between the two, and so we'll check this assumption off for now.

Homoscedasticity

Looking at the plot we just made, I think we may have found our problem - the variance in both variables is much higher at high values (our data forms a cone-shape around the regression line). We can see this very directly by plotting the residuals vs. our predicted ("fitted") values:

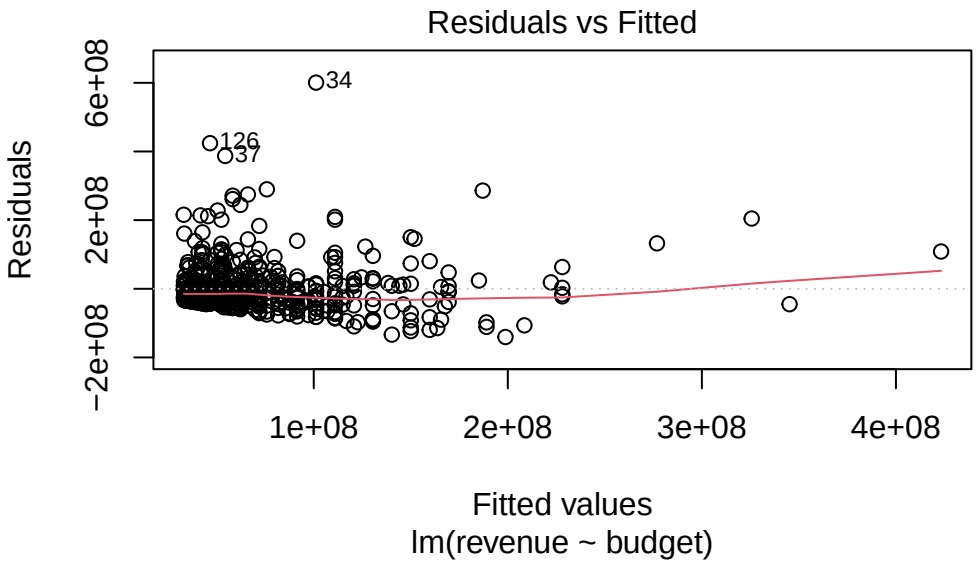
```
res_df <- data.frame(residuals = rvb_lm$residuals,
                     fitted.values = rvb_lm$fitted.values)

res_df %>%
  ggplot(aes(x = fitted.values, y = residuals)) +
  geom_point()
```

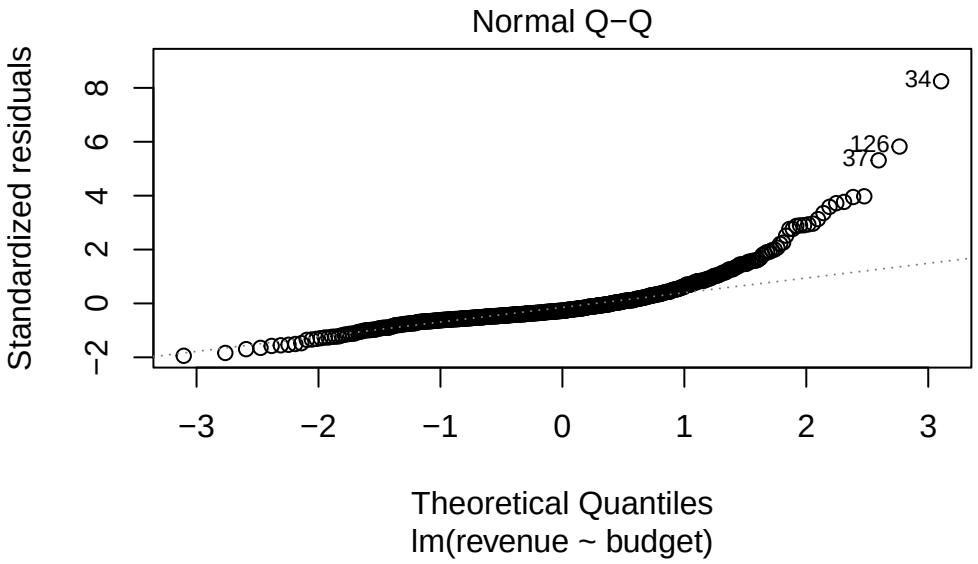


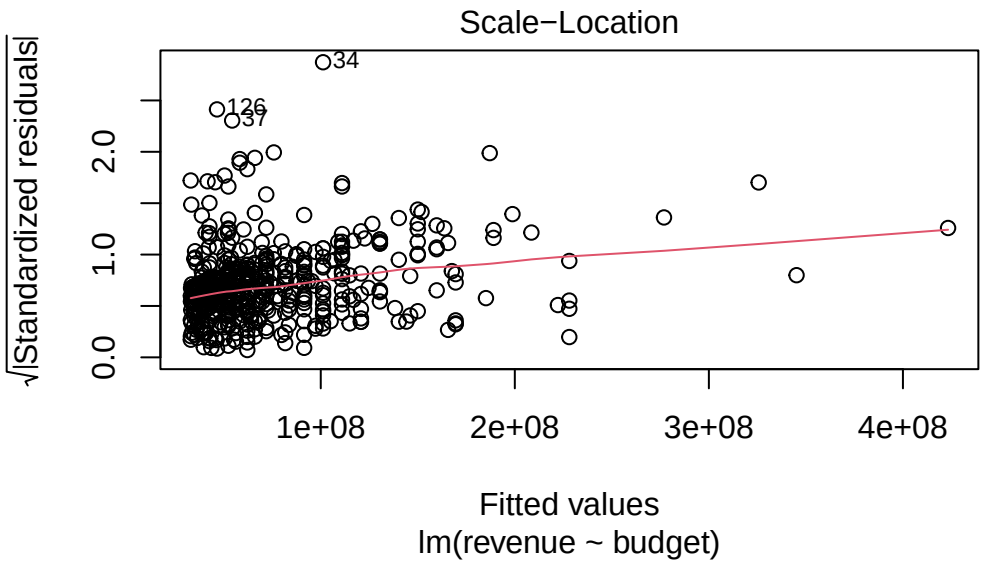
Hmmm. We want these to be evenly distributed around 0. The residuals are actually more variable at lower budgets.

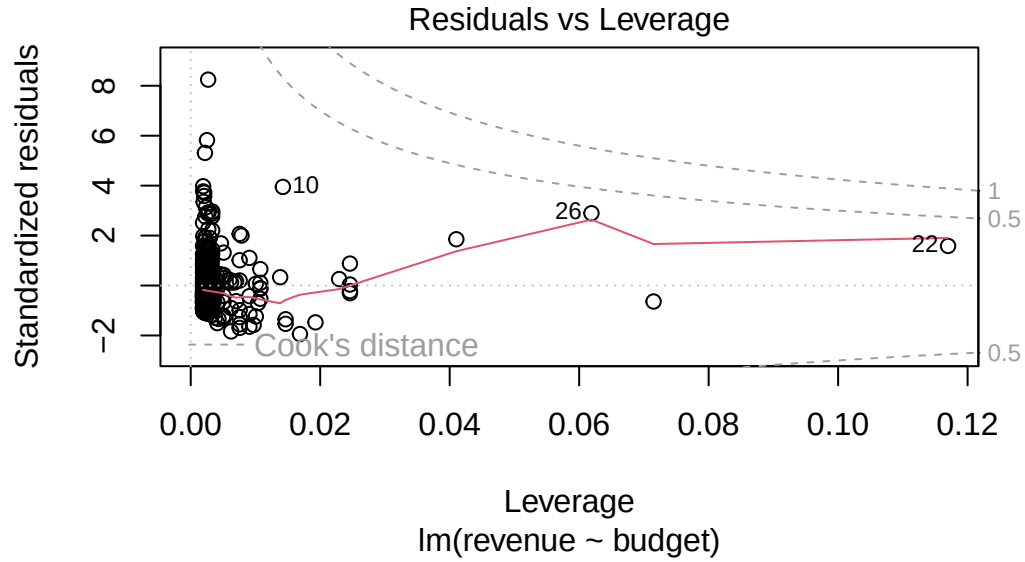
Fun fact - we can get this plot directly from our linear model using the plot() function:

```
plot(rvb_lm)
```


Residuals vs Fitted
Residuals
Fitted values
lm(revenue ~ budget)
34
126
37
6e+08
2e+08
−2e+08
1e+08
2e+08
3e+08
4e+08


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(revenue ~ budget)
34
126
37
8
6
4
2
0
−2
−3
−2
−1
0
1
2
3





Normally distributed residuals

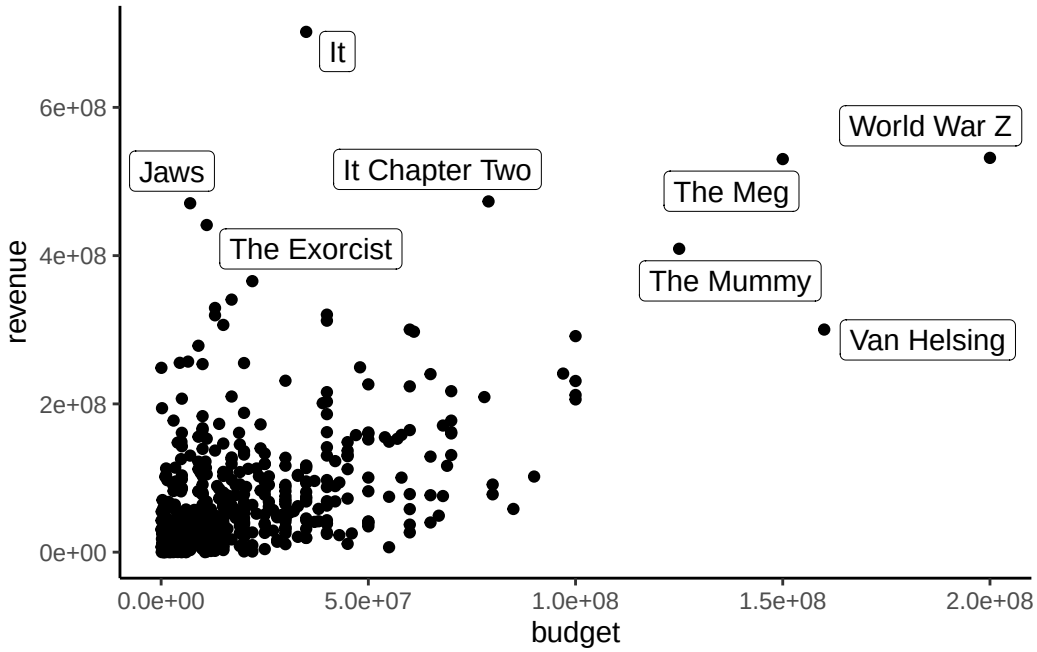
Luckily, we've already plotted the residual distribution! It was pretty normal, but we saw a

fairly significant right-skew.

Outliers

It might be a good idea to look at our data again to look for outliers. I'm looking especially at our points that had extremely high revenues or very high budgets, so I'm labeling them with the geom_label_repel() function from the ggrepel library:

```
horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  ggrepel::geom_label_repel(aes(label = title), label.size = 0)
```



Clearly, there are some relatively low budget movies like "Jaws" and "It" which had huge amounts of revenue. Are these outliers though? Is there any valid reason for us to remove them?

Probably not! For one, we don't think there was a technical error during collection of these

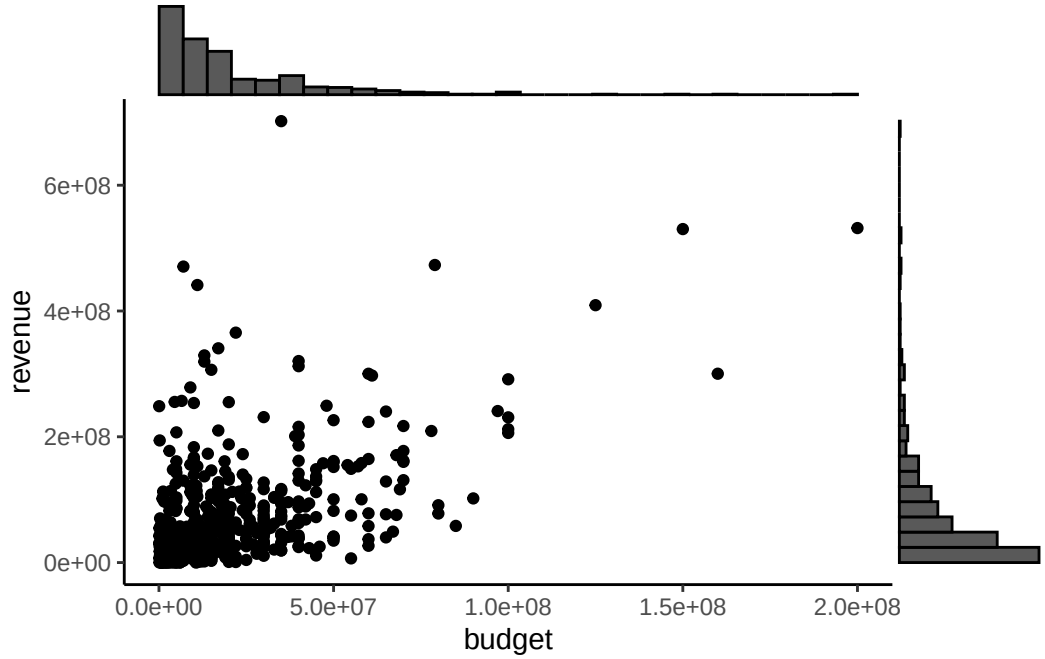
data. And if we look at the Residuals vs. Leverage plot, none of our points fall outside of Cook's Distance lines, which would indicate a single point is strongly influencing our model. Sure, that means our model will have a worse R-squared value. But, we want to be realistic, and recognize that of course a linear model will almost never perfectly describe phenomenon in the outside world.

Trying a transformation:

Our model seems to be suffereing because we're observing heterscedasticity in the variance of revenue across budget. Maybe what we're realizing is that it's possible for low-budget movies to become smash hits, but high-budget movies are constrained - there's only so much money they can recover from a genre movie, and they also can't really flop because of studio support.

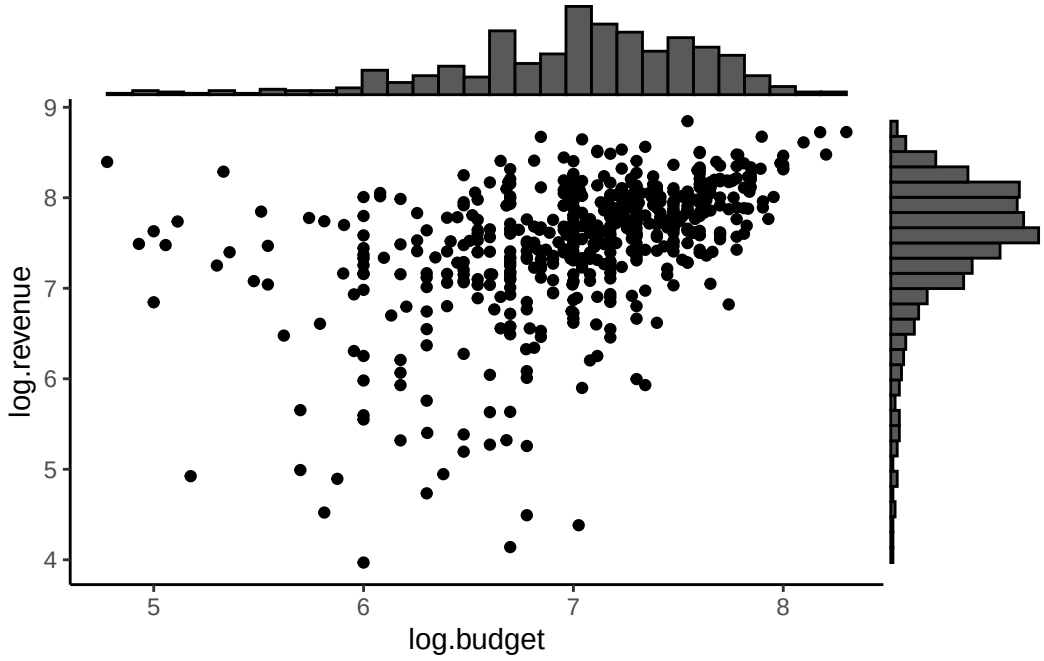
There might still be better ways for us to model this data. One method is to **transform** our data, which may help us deal with this heteroscedasticity. Let's look at the distributions of our variables. I'm using a really cool library called ggExtra to add histograms of each variable to our ggplot:

```
plot <- horror_data%>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point()
ggExtra::ggMarginal(plot, type = "histogram")
```



As we can see, neither of our distributions are normally distributed **at all**. This does not necessarily mean they aren't fit for linear regression - our assumption is normality of the residuals, not the variables themselves. However, a transformation can help us even out error when our data is strongly skewed. Let's transform our data and look at this again:

```
log_horror_data <- horror_data %>%
  mutate(log.revenue = log10(revenue),
         log.budget = log10(budget))
log.plot <- log_horror_data %>%
  ggplot(aes(x = log.budget, y = log.revenue)) +
  geom_point()

ggExtra::ggMarginal(log.plot, type = "histogram")
```



Okay - now our data are left-skewed, sure, but they are much more normal than they were before! Let's build a linear model using our log-transformed data:

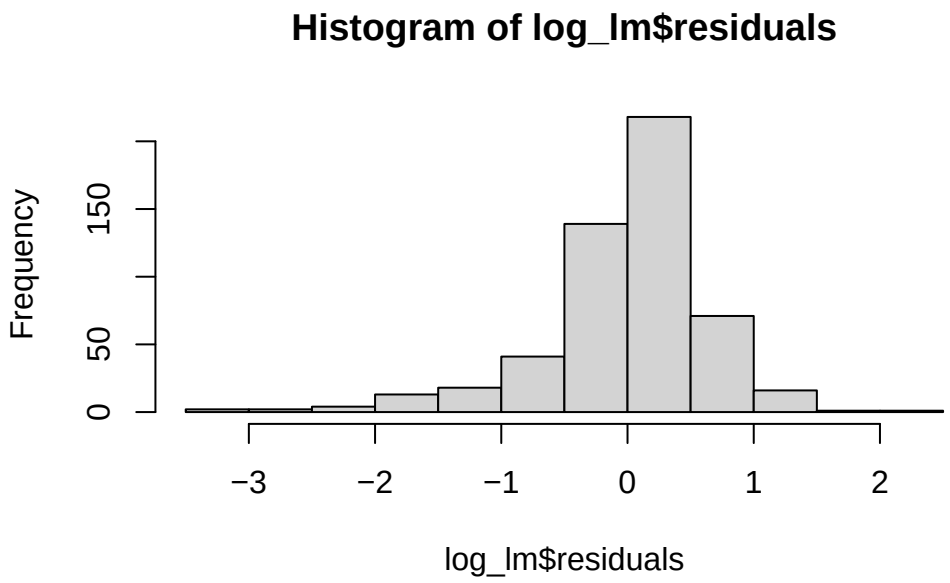
```
log_lm <- lm(log.revenue~log.budget, data = log_horror_data)

summary(log_lm)
```

```
Call:
lm(formula = log.revenue ~ log.budget, data = log_horror_data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1742 -0.2440  0.0935  0.3647  2.2074

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.38677    0.34964   9.686   <2e-16 ***
log.budget   0.58630    0.04967  11.803   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6683 on 524 degrees of freedom
Multiple R-squared:   0.21, Adjusted R-squared:  0.2085
F-statistic: 139.3 on 1 and 524 DF,  p-value: < 2.2e-16
```

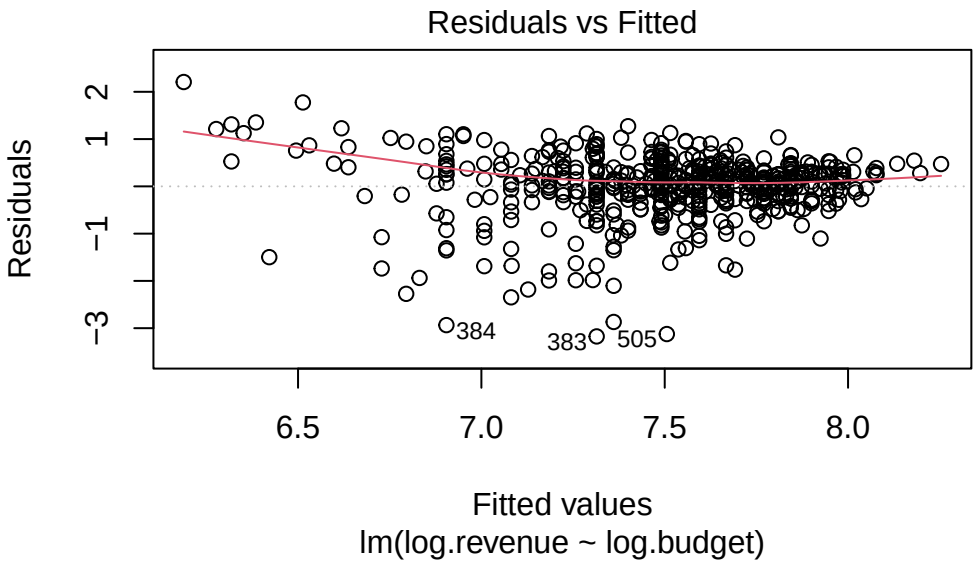
Okay - we can't compare the residual standard error to our original data, because of the log transformation (it would be a mistake to take $10^{0.66}$ and call it good). But we can look at R-squared, which is unitless. We've actually lost some of our predictive power with this model! But is the model "worse"? Let's look at our diagnostic plots:

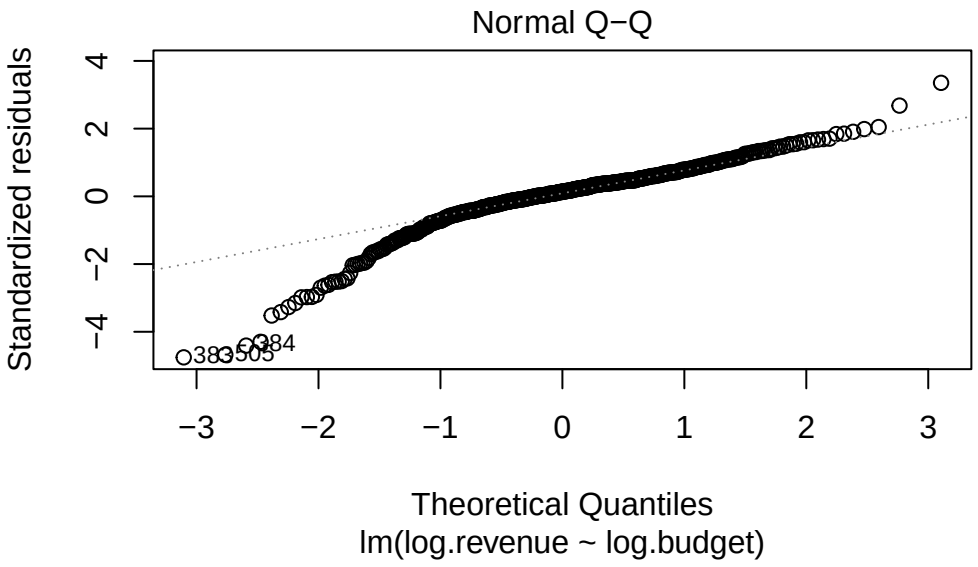
```
hist(log_lm$residuals)
```

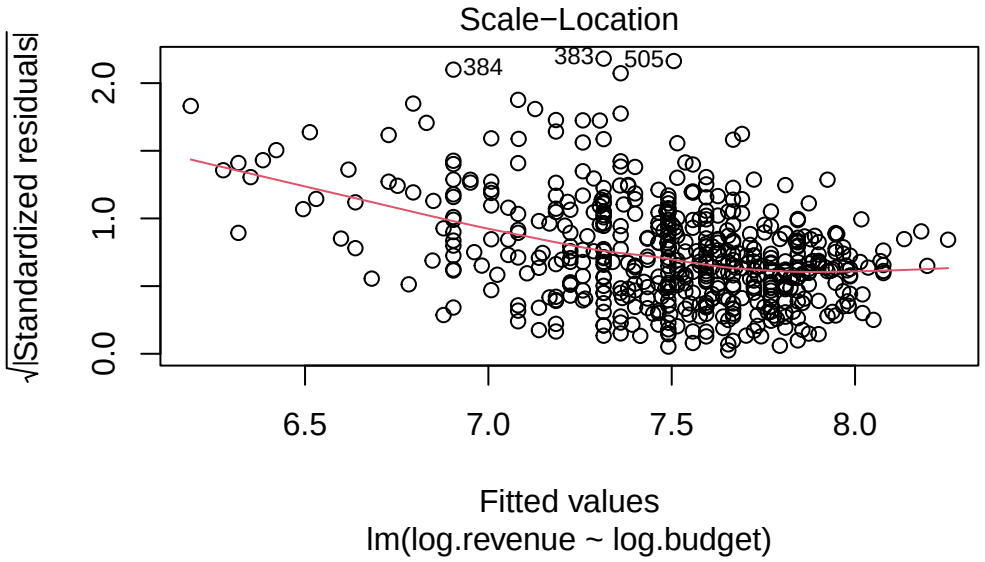


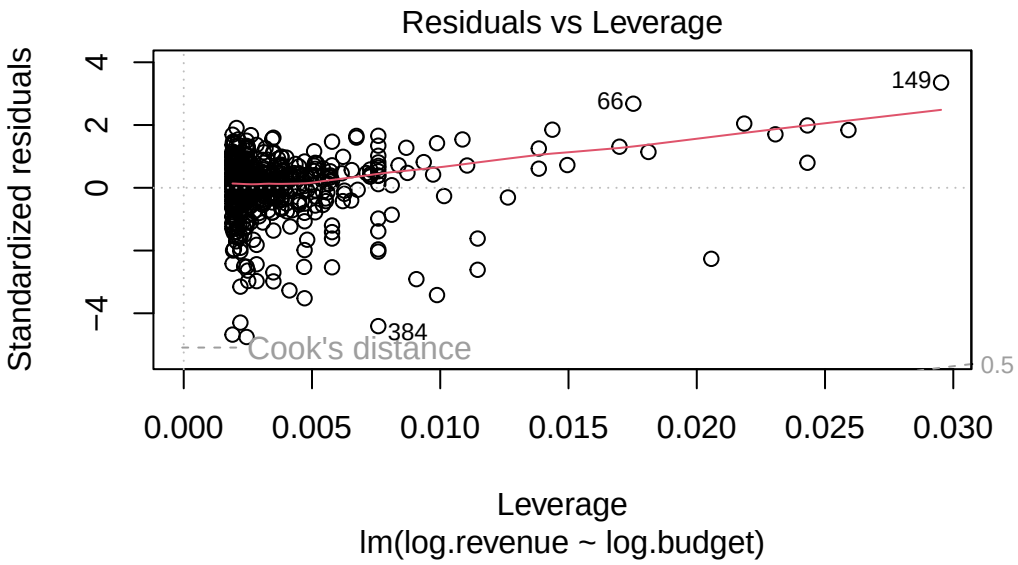
Our residuals look a little more normal.

```
plot(log_lm)
```



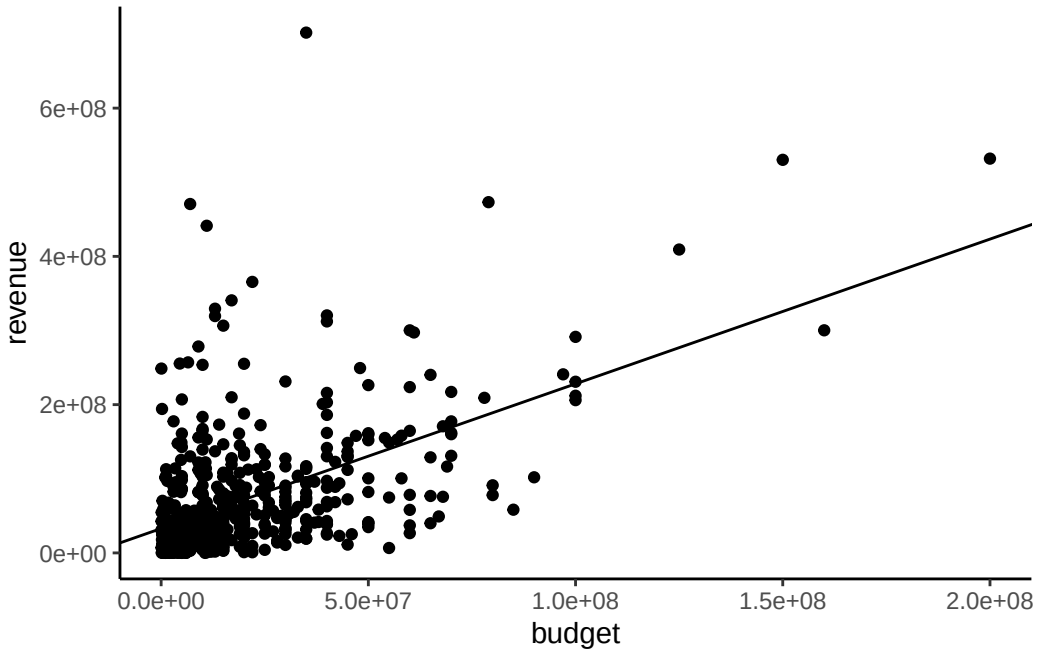

Normal Q−Q
Standardized residuals
4
2
0
−2
−4
383
505
384
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(log.revenue ~ log.budget)


Scale−Location
√|Standardized residuals|
2.0
1.0
0.0
384
383
505
6.5
7.0
7.5
8.0
Fitted values
lm(log.revenue ~ log.budget)

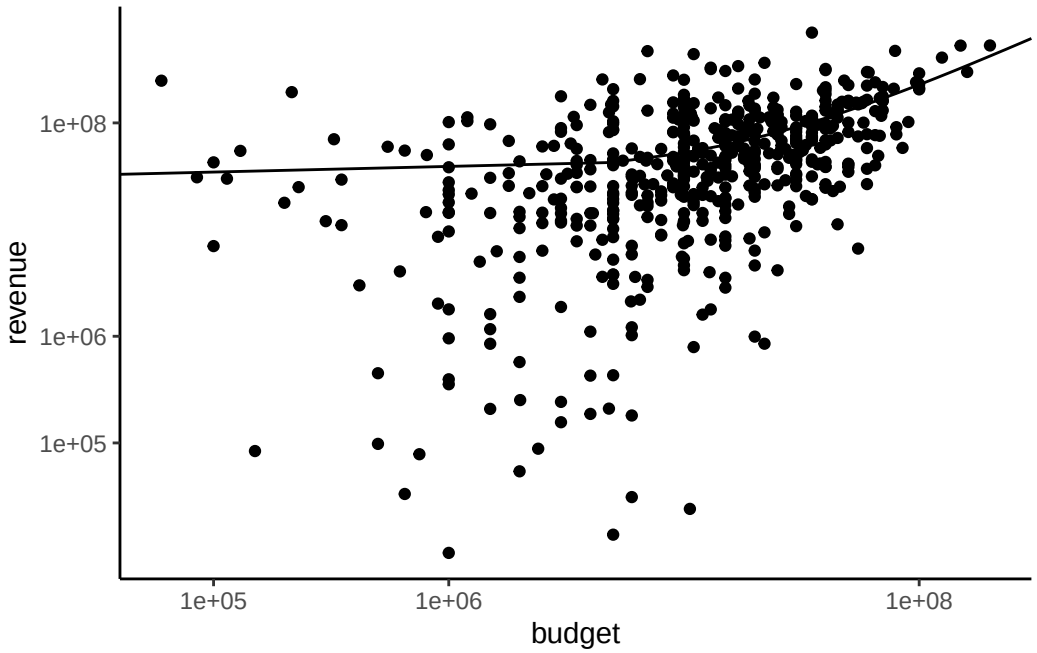


The residuals look a little more evenly spread across our fitted values. However, we are still seeing a bigger variance in residuals at lower budgets, and a consistent trend of more positive residuals at lower budgets. The Q-Q plot also shows potentially worse deviation from a normal deviation than in our original model. As such, maybe I'll stick with my original linear model! For plotting, it may be easier to visualize these data on a log-scale though (our linear model will look curved in log-space). I don't know, what do you think?

```
# Plot in linear space
horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  geom_abline(intercept = rvb_lm$coefficients["(Intercept)"], slope = rvb_lm$coefficients[
```




```
# Plot in log space
horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  geom_abline(intercept = rvb_lm$coefficients["(Intercept)"], slope = rvb_lm$coefficients[
  scale_x_continuous(breaks = c(10^5,10^6, 10^8))+
  scale_y_continuous(breaks = c(10^5, 10^6, 10^8))+
  coord_trans(x = "log10", y = "log10")
```



Summary

In today's lesson, we learned how to construct a linear model in R, understand R's linear model output, and assess the quality of our linear model. We've learned there's not a single one thing that makes a model "good", and we should use our knowledge of the problem and our common sense when interpreting our results.

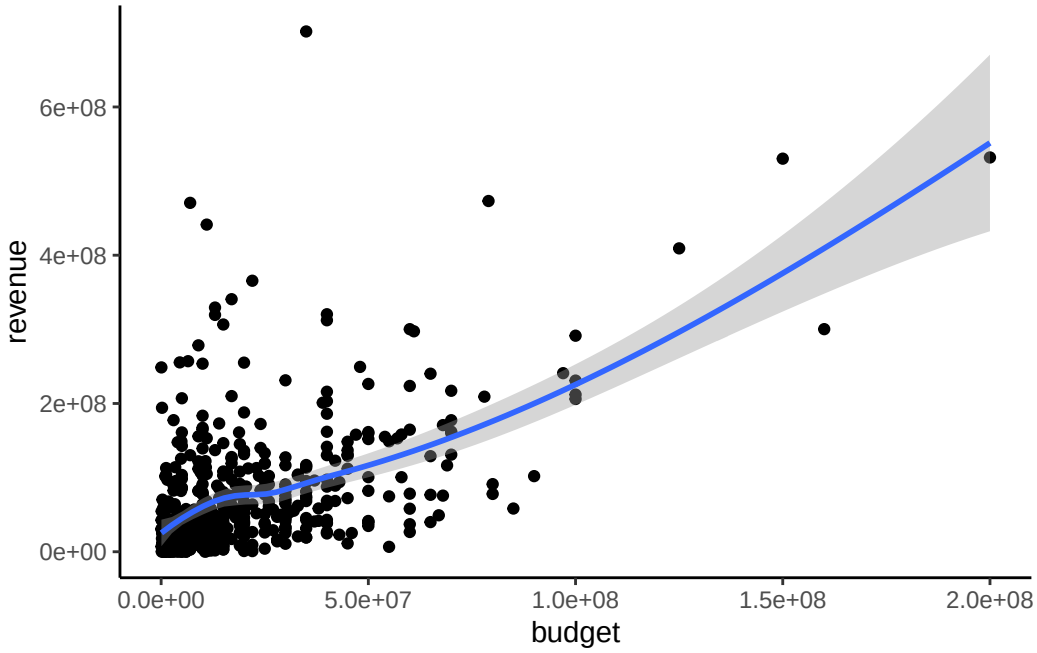
Bonus: Adding linear models to ggplots "on-the-fly"

As good statisticians, we always read the entire output from our linear models and look at the diagnotic plots to assess the quality of our models, right? But maybe sometimes, rarely, we just want to slap a line onto a ggplot and call it good. There are a few ways to do this.

The first way uses the geom_smooth() function from ggplot2:

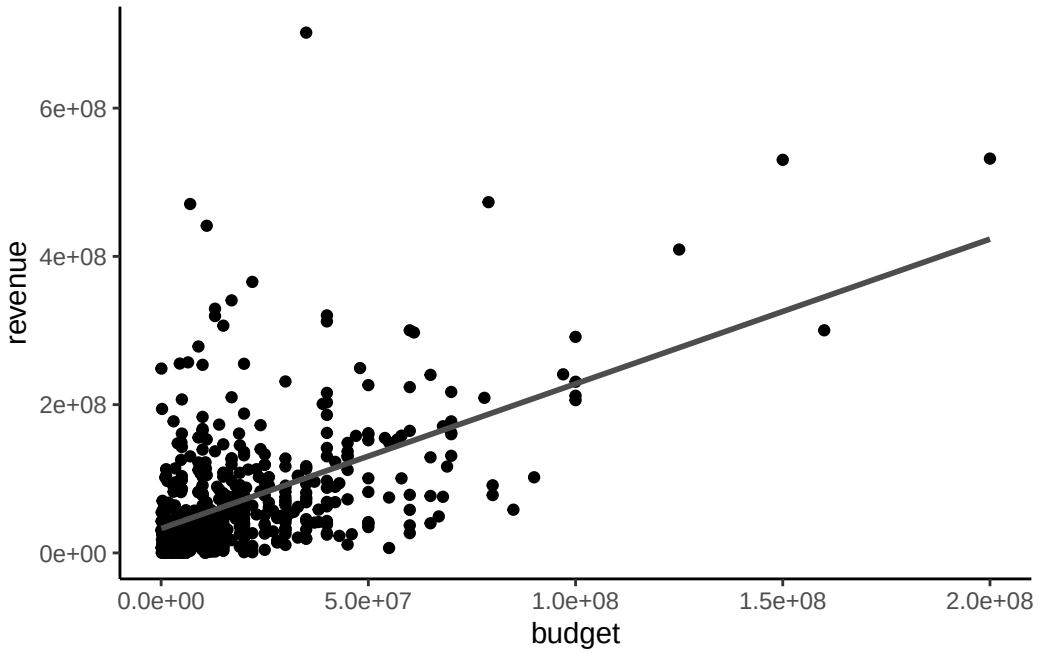
```
horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
```

```
  geom_smooth()
```



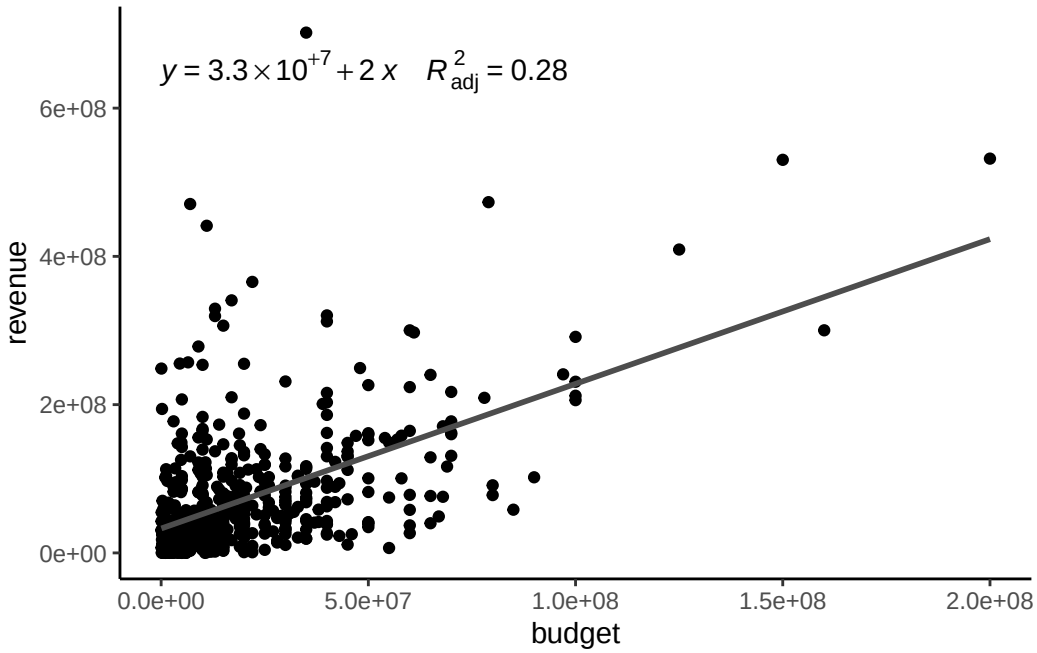
A trend line is added to the plot in blue, and there is a gray 95% confidence interval added as well. By default, ggplot2 fits what's called a "loess" model, which is a bunch of small linear models in a trenchcoat and gives us this curving line. In my personal view, this often risks overfitting your data and isn't particularly informative, so let's change the method to a linear model ("lm"). We can also change the color and get rid of the 95% confidence interval:

```
horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "grey30")
```



Sometimes, we also want to see information from our model like the coefficients and the R-squared (Excel makes this very easy). We could either pull those data out of our linear model and "annotate" them onto our graph manually, or use functions in the ggpubr library to do it automatically:

```
# Manually annotate:


horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "grey30") +
  ggpubr::stat_regline_equation(
    aes(label = paste(after_stat(eq.label), after_stat(adj.rr.label),
                      sep = "~~~~")))
```



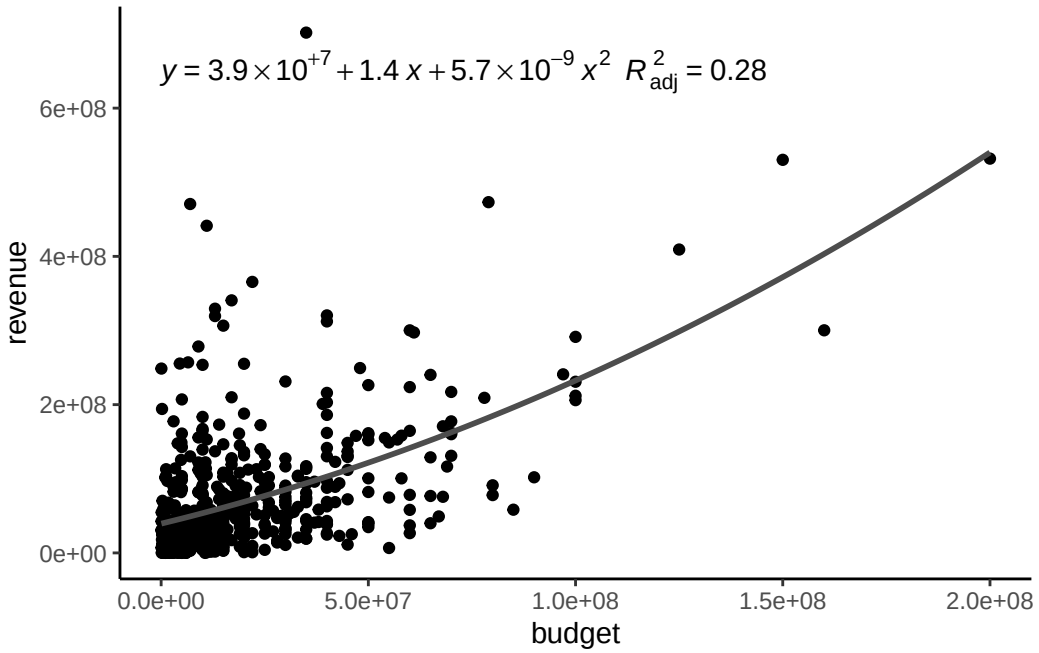
We can do this even more more complex formulas (here, a quadratic formula):

```
lm(revenue~poly(budget, 2, raw = T), data = horror_data)%>%summary()
```

```
Call:
lm(formula = revenue ~ poly(budget, 2, raw = T), data = horror_data)

Residuals:
       Min          1Q      Median          3Q         Max
-137974791  -39882606  -20291898   18417231  607845585

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 3.945e+07  4.969e+06   7.938 1.26e-14 ***
poly(budget, 2, raw = T)1 1.358e+00  2.770e-01   4.903 1.26e-06 ***
```

```
poly(budget, 2, raw = T)2 5.727e-09  2.323e-09   2.466    0.014 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 72590000 on 523 degrees of freedom
Multiple R-squared:  0.2859,    Adjusted R-squared:  0.2832
F-statistic: 104.7 on 2 and 523 DF,  p-value: < 2.2e-16
```

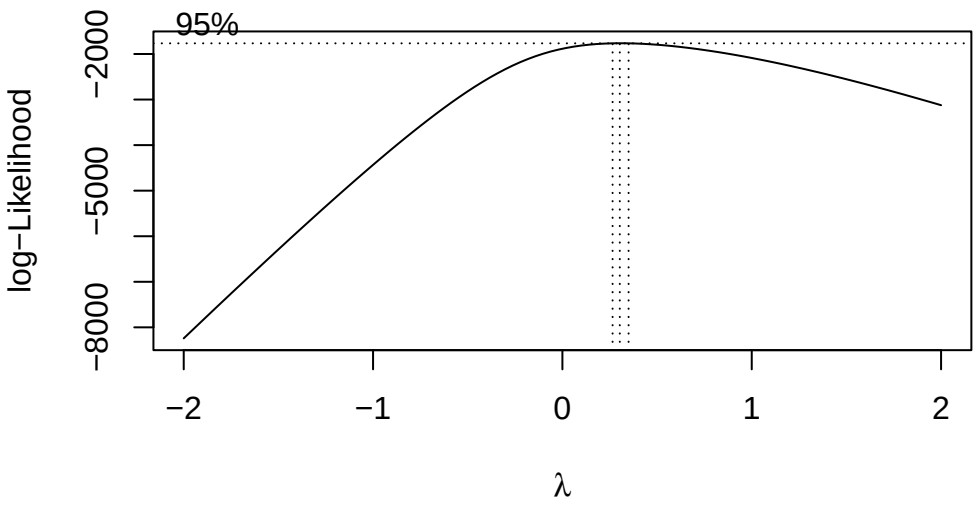
```
horror_data %>%
  ggplot(aes(x = budget, y = revenue)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "grey30",
              formula = y~poly(x, 2, raw = TRUE)) +
  ggpubr::stat_regline_equation(
    aes(label = paste(after_stat(eq.label), after_stat(adj.rr.label),
                      sep = "~~")),
                      formula = y~poly(x, 2, raw = TRUE))
```



Bonus: Box-Cox Transformations

Sometimes, the most common transformation approaches (log, power, inverse) don't quite normalize our data's variance. The Box Cox transformation is a method which transforms your data across a range of "lambda" to find the most normal result, essentially testing all possible transformation methods. We can do this in R using the following code:

```
library(MASS) # Load the library with the boxcox function


bc_lm <- boxcox(rvb_lm) # Feed the boxcox function our linear model
```

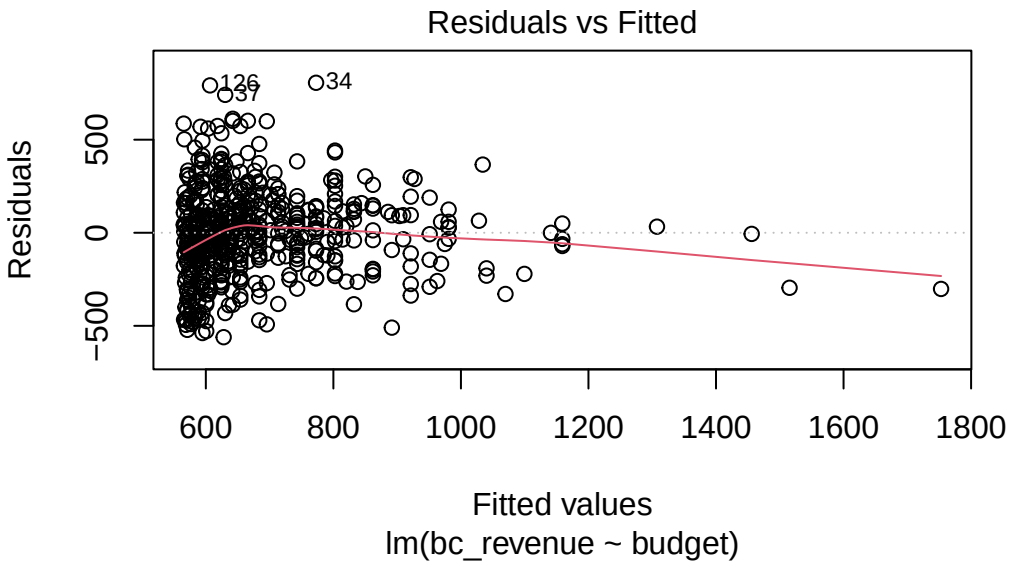


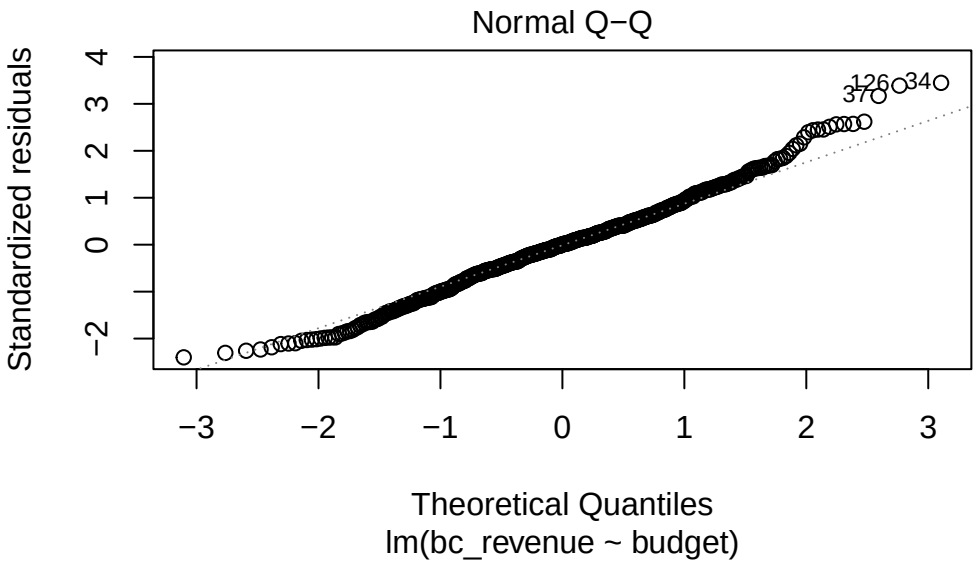

```
lambda <- bc_lm$x[which.max(bc_lm$y)] # Find the best value of lambda (x)

bc_horror_data <- horror_data%>%
  mutate(bc_revenue = (revenue^lambda - 1)/lambda) # Transform our data

bc_lm <- lm(bc_revenue~budget, data = bc_horror_data)

summary(bc_lm)
```

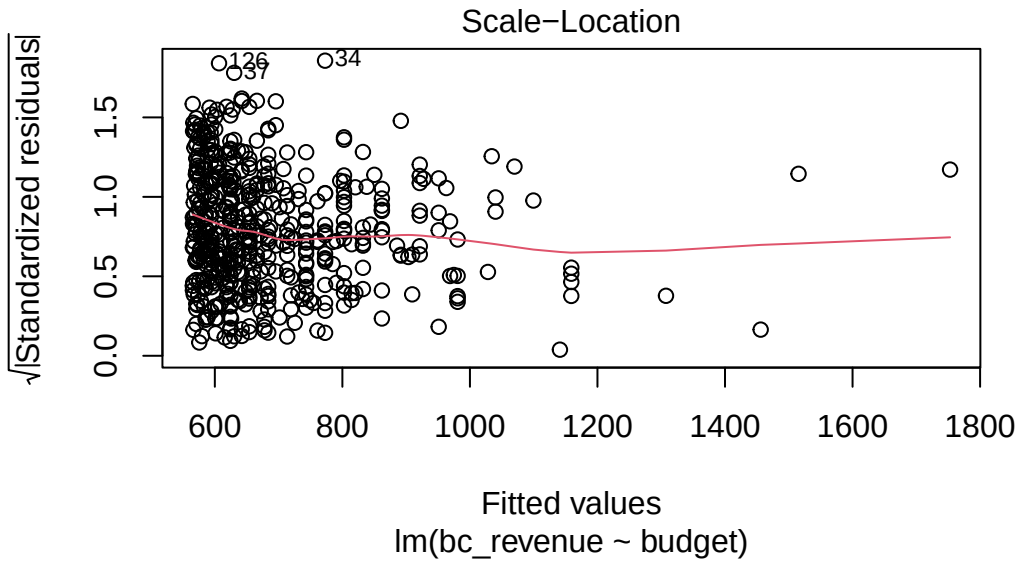
```
Call:
lm(formula = bc_revenue ~ budget, data = bc_horror_data)

Residuals:
    Min      1Q  Median      3Q     Max
-561.04 -141.91    1.85  136.68  805.79
```

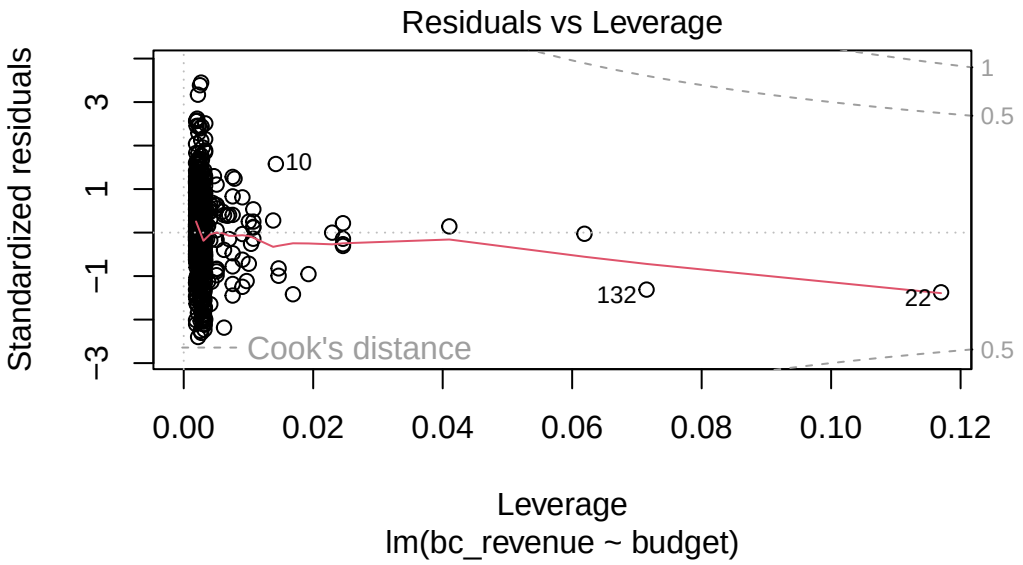
```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.650e+02  1.353e+01   41.77   <2e-16 ***
budget      5.940e-06  4.413e-07   13.46   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 234 on 524 degrees of freedom
Multiple R-squared:  0.2569,    Adjusted R-squared:  0.2555
F-statistic: 181.2 on 1 and 524 DF,  p-value: < 2.2e-16
```

```r
plot(bc_lm)
```




Normal Q−Q
Standardized residuals
126
34
37
−2
0
1
2
3
4
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(bc_revenue ~ budget)


Scale−Location
√|Standardized residuals|
126
37
34
0.0
0.5
1.0
1.5
600
800
1000
1200
1400
1600
1800
Fitted values
lm(bc_revenue ~ budget)



The Q-Q plot looks the best we've seen, so we have in fact transformed towards more normal residuals. But we still have a worse R-squared than our original model! We would need to think critically about what's more important in our research context to decide the best model. We could also look at other quality metrics, like RMSE, to help us decide.