# PSTAT 126 - Lab 3

Fall 2022

*Note: If you are having trouble with Rstudio or Rmarkdown, come to office hours or message us on Nectir.*

## Dataset for today: Baseball!

### Installing Data Package

A package named `Lahman` contains baseball data. To install the package, running the following command in the RStudio console:

```
install.packages('Lahman')
```

See the description of the data package by running,

```
help("Lahman-package", package="Lahman")
```

Following lists the datasets:

```
help(package="Lahman")
```

Load the data package

```
library(tidyverse)
library(Lahman) # Package with baseball (MLB) statistics
str(Batting)
```

```
## 'data.frame':    110495 obs. of  22 variables:
##  $ playerID: chr  "abercda01" "addybo01" "allisar01" "allisdo01" ...
##  $ yearID  : int  1871 1871 1871 1871 1871 1871 1871 1871 1871 1871 ...
##  $ stint   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ teamID  : Factor w/ 149 levels "ALT","ANA","ARI",..: 136 111 39 142 111 56 111 24 56 24 ...
##  $ lgID    : Factor w/ 7 levels "AA","AL","FL",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ G       : int  1 25 29 27 25 12 1 31 1 18 ...
##  $ AB      : int  4 118 137 133 120 49 4 157 5 86 ...
##  $ R       : int  0 30 28 28 29 9 0 66 1 13 ...
##  $ H       : int  0 32 40 44 39 11 1 63 1 13 ...
##  $ X2B     : int  0 6 4 10 11 2 0 10 1 2 ...
##  $ X3B     : int  0 0 5 2 3 1 0 9 0 1 ...
##  $ HR      : int  0 0 0 2 0 0 0 0 0 0 ...
##  $ RBI     : int  0 13 19 27 16 5 2 34 1 11 ...
##  $ SB      : int  0 8 3 1 6 0 0 11 0 1 ...
##  $ CS      : int  0 1 1 1 2 1 0 6 0 0 ...
##  $ BB      : int  0 4 2 0 2 0 1 13 0 0 ...
##  $ SO      : int  0 0 5 2 1 1 0 1 0 0 ...
##  $ IBB     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ HBP     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ SH      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ SF      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ GIDP    : int  0 0 1 0 0 0 0 1 0 0 ...
```

## Model 1

- Can we predict the number of Runs by the number of At Bats?

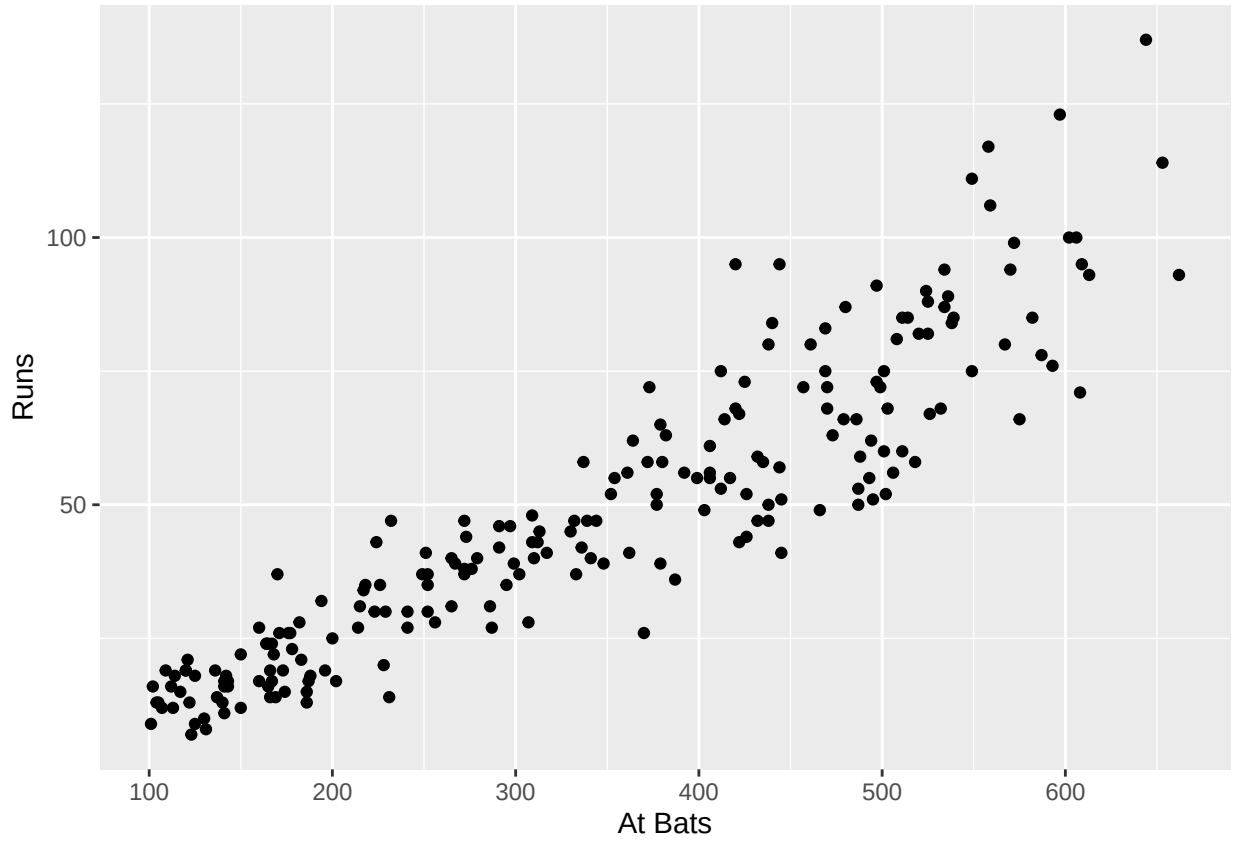
```
df1 <- Batting %>%
filter(yearID == "2017" & # stats from 2017
        lgID == "NL" & # only from the NL
        AB > 100) # only At Bats more than 100.

dim(df1)
```

```
## [1] 215  22
```

**Remark:** to read information about the data set, type `?Batting` into the Console. Here we are restricting the data set – we are only looking at statistics from players in 2017 that played in the the National League (NL) and went up to bat at least 100 times. You can see this restricts us to 215 observations.

```
p <- df1 %>% ggplot()+
  geom_point(aes(x=AB, y=R))+
  labs(x = "At Bats", y = "Runs")
p
```




Looks pretty linear! Lets use the `lm` function to fit the model

$$\text{Runs}_i = \beta_0 + \beta_1 \cdot \text{At Bats}_i + \epsilon_i.$$

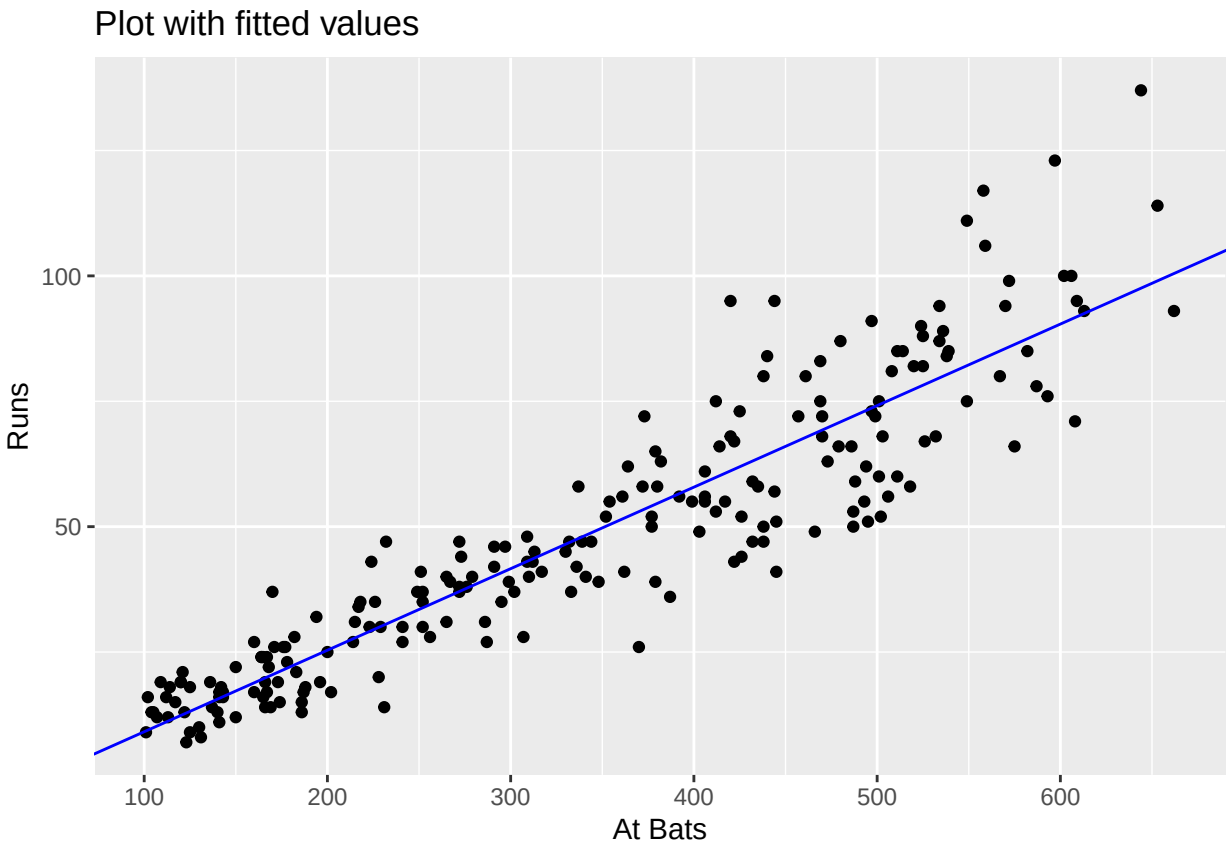
```
Runs <- df1$R
At_Bats <- df1$AB
```

```
model_Runs <- lm(Runs ~ At_Bats, data = df1)
summary(model_Runs)
```

```
##
## Call:
## lm(formula = Runs ~ At_Bats, data = df1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.001  -5.851   0.231   5.554  39.452
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.154968   1.807307  -3.959 0.000103 ***
## At_Bats      0.162583   0.004844  33.562  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.89 on 213 degrees of freedom
## Multiple R-squared:  0.841,  Adjusted R-squared:  0.8402
## F-statistic:  1126 on 1 and 213 DF,  p-value: < 2.2e-16
```

We can add our model function to our plot from before

```
p + geom_abline(aes(intercept = model_Runs$coefficients[1],
                    slope = model_Runs$coefficients[2]),
                color = "blue") +
  ggtitle("Plot with fitted values")
```




## Simple Linear Regression Model Assumptions

1) The mean response is a linear function of $x_i$. **L**inearity
2) Errors have **E**qual variance. $\text{Var}(Y_i) = \sigma^2$ for every $i$
3) Errors are **N**ormally distributed
4) Errors are **I**ndependent

- Can use the acronym **L.I.N.E.** to help you remember.

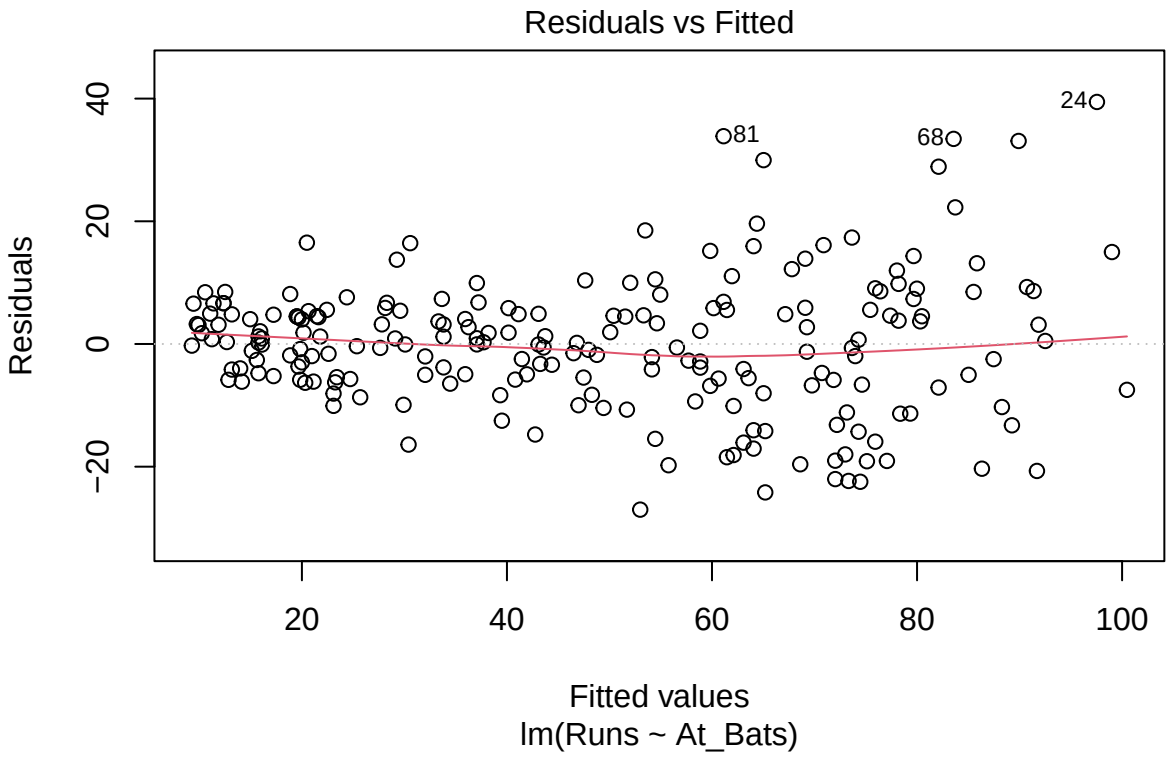
### How do we test that these assumptions hold?

- Linearity = Residuals vs. Fitted plot
- Constant Variance = Scale - Location
- Normality = QQ plot
- Independence = Complicated. Requires knowledge of study design or data collection in order to establish the validity of this assumption. No general test.

For the first 3 assumptions, using the base R `plot` function on an `lm` object generates 3 plots which we can respectively use to assess if the assumptions hold.

### Linearity

```
plot(model_Runs, which = 1)
```




We are looking for there to be no obvious pattern around the horizontal line. The residuals are more or less evenly distributed above and below the line.

Our assumption is that "the mean response is a linear function of $x_i$"

$$E[Y_i] = E[\beta_0 + \beta_1 x_i + \epsilon_i] = \beta_0 + \beta_1 x_i + \underbrace{E[\epsilon_i]}_{=0}$$

Since the residuals $e_i$ are an approximation to our errors $\epsilon_i$, if the residuals are evenly spread around the horizontal line, this gives us evidence that the errors have zero mean.

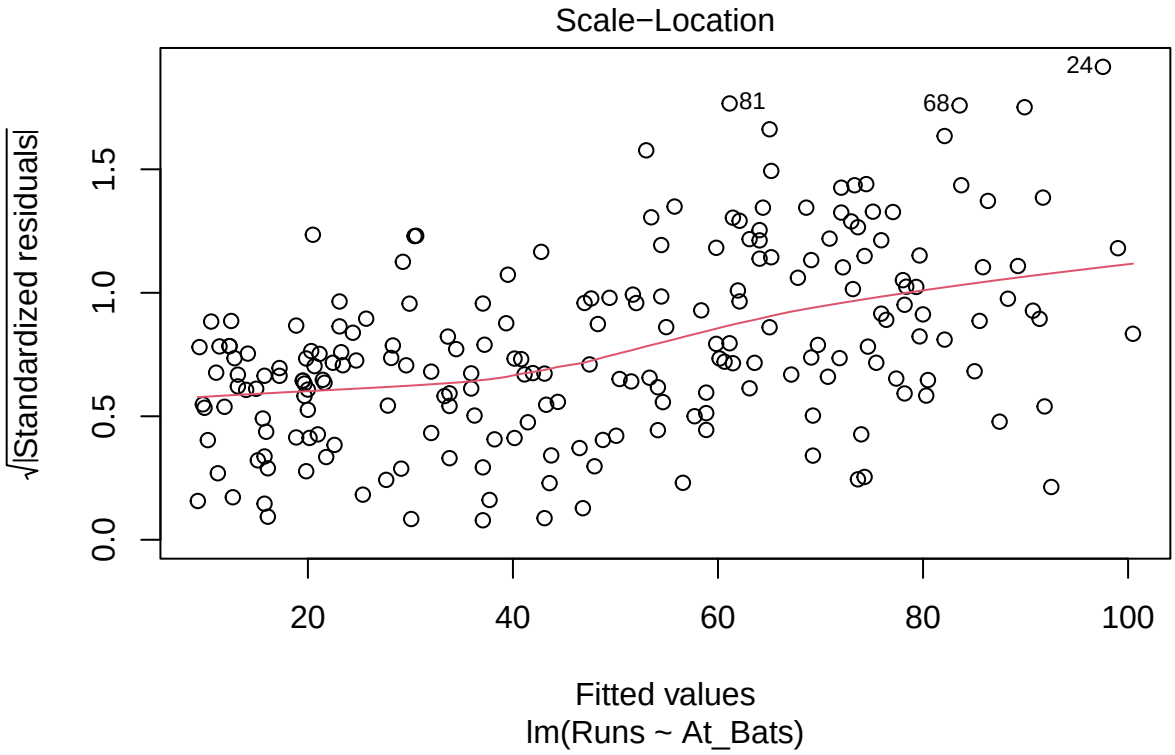
This overall really amounts to the assumption that our model specification is correct.

**Constant Variance**

Many times, we can analyze whether our residuals have constant variance also by looking at the Residuals vs. Fitted Values plot above. Scanning left to right, if the points seem the band of points seems to have a constant width, that's what we want.

However, a Scale-Location plot just makes it easier to assess homoscedacity issues.

```
plot(model_Runs, which = 3)
```




The red line gives a measure of the average magnitude of the residuals around that fitted value. If the residuals have constant variance, we would expect the red line to be flat.

The Scale-Location is particularly easier to analyze than the Residual plot when the data is more sparse or when the points are rather unevenly distributed along the $x$-axis.

In our example, the red line has an upward trend, indicating the variance of the residuals increases as the fitted values increase.

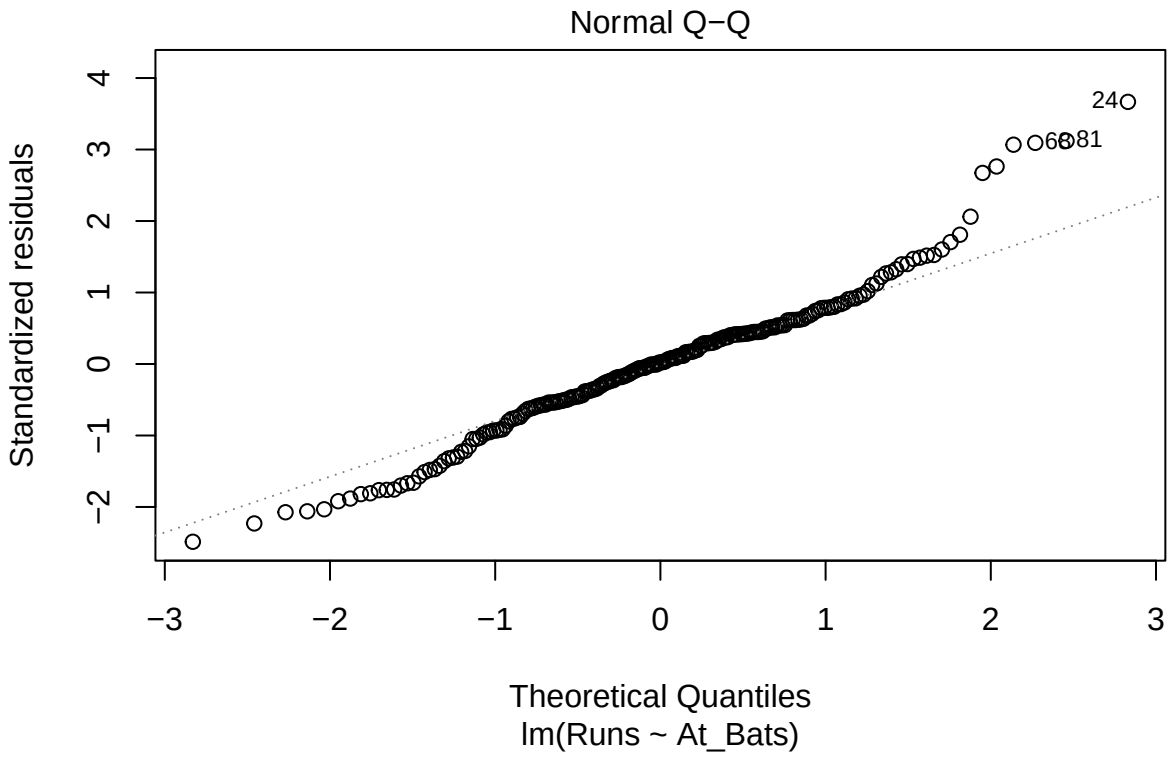
A common trick to attempt to vanquish heteroscedacity is to transform the response variable using a transform such as a log or square root.

**Normality**

This assumption is arguably the least important. The normality of the errors (and thus the residuals) is used to deduce the distributions of our OLS estimators, and with this knowledge, we can justify our confidence intervals and significance tests for our estimates.

Furthermore, non-normality may only be of concern with small sample sizes, otherwise, the Central Limit Theorem will ensure normality of our residuals.

```
plot(model_Runs, which = 2)
```




With a QQ-plot, we are looking for all the points to line on the line to say that residuals are normally distributed.

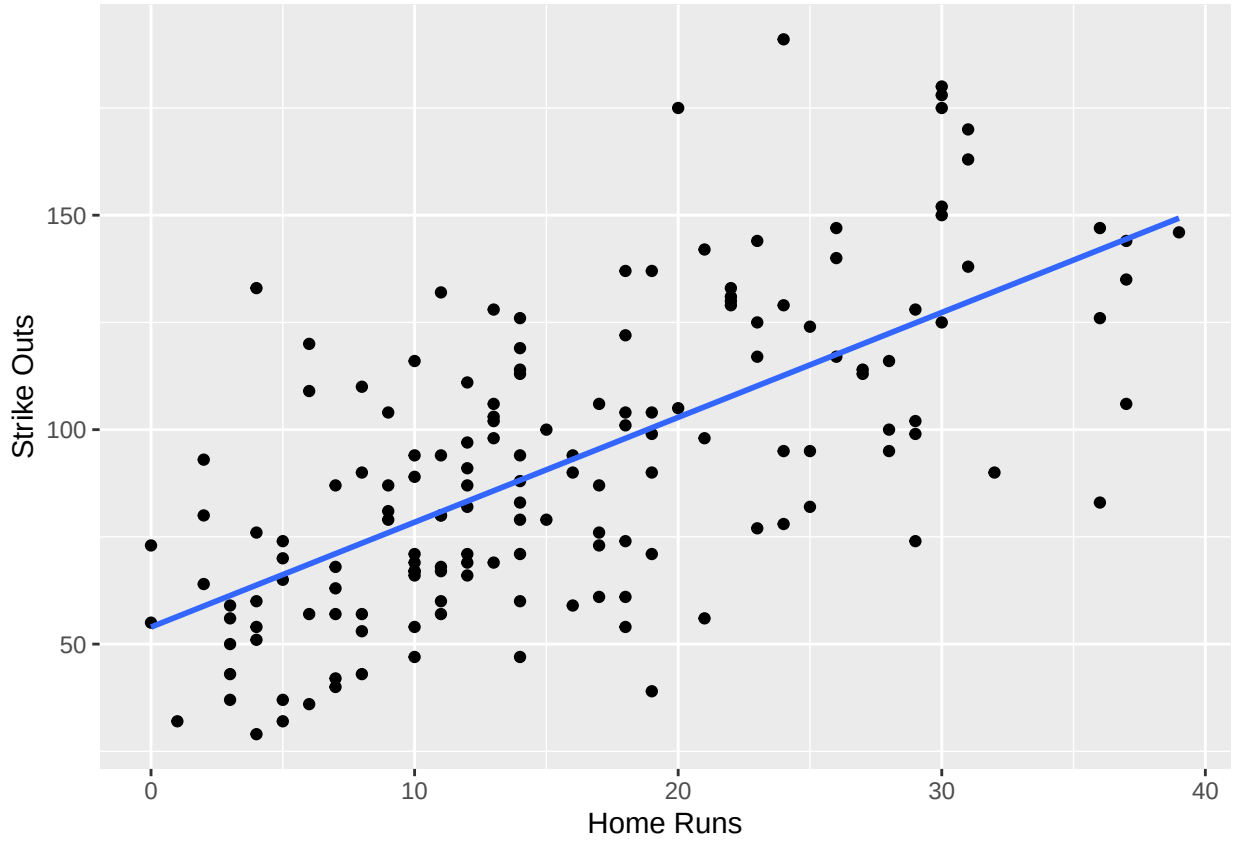
## Model 2

- Can we predict Strike outs by Home runs?

```
df2 <- Batting %>%
filter(yearID == "2017" & # stats from 2017
        lgID == "NL" & # only from the NL
        AB > 200 & # Players who have more than 200 At Bats
        HR < 45) # Players who have less than 45 home runs
dim(df2)
```

```
## [1] 156  22
```

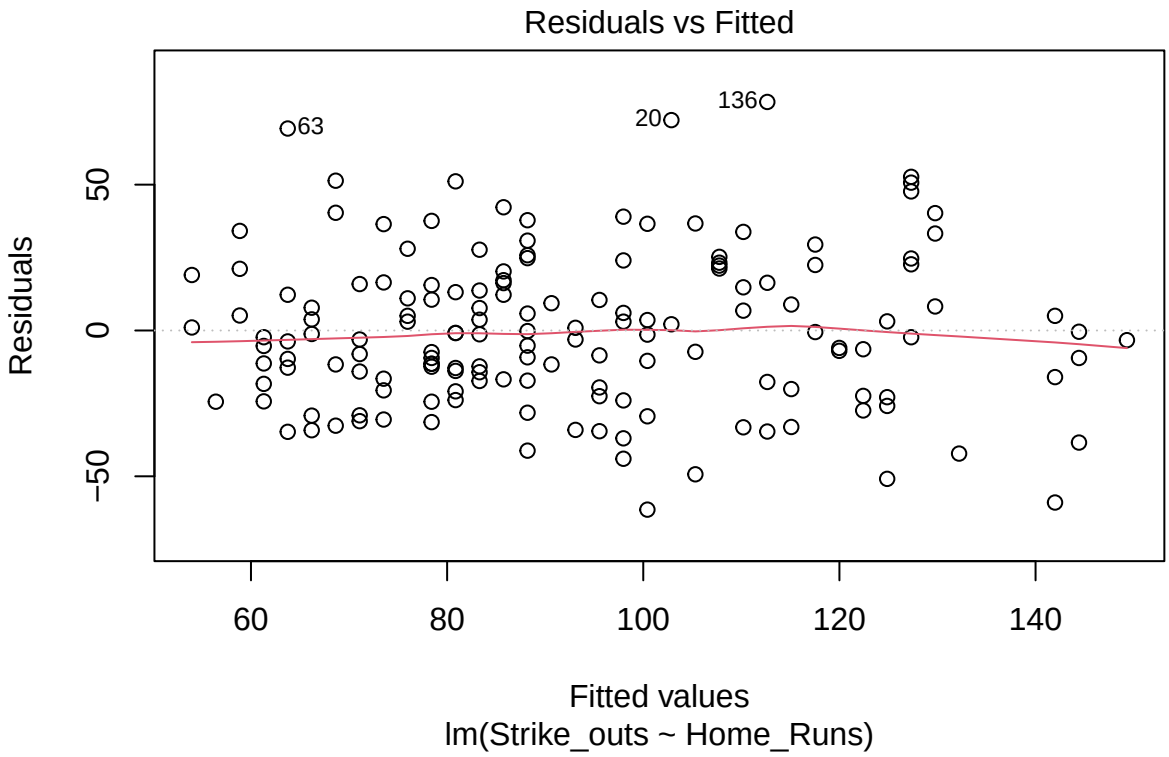
```
df2 %>% ggplot(aes(x = HR, y = SO))+
  geom_point()+
  geom_smooth(method = "lm", formula = "y~x", se = F)+
  labs(x="Home Runs", y="Strike Outs")
```

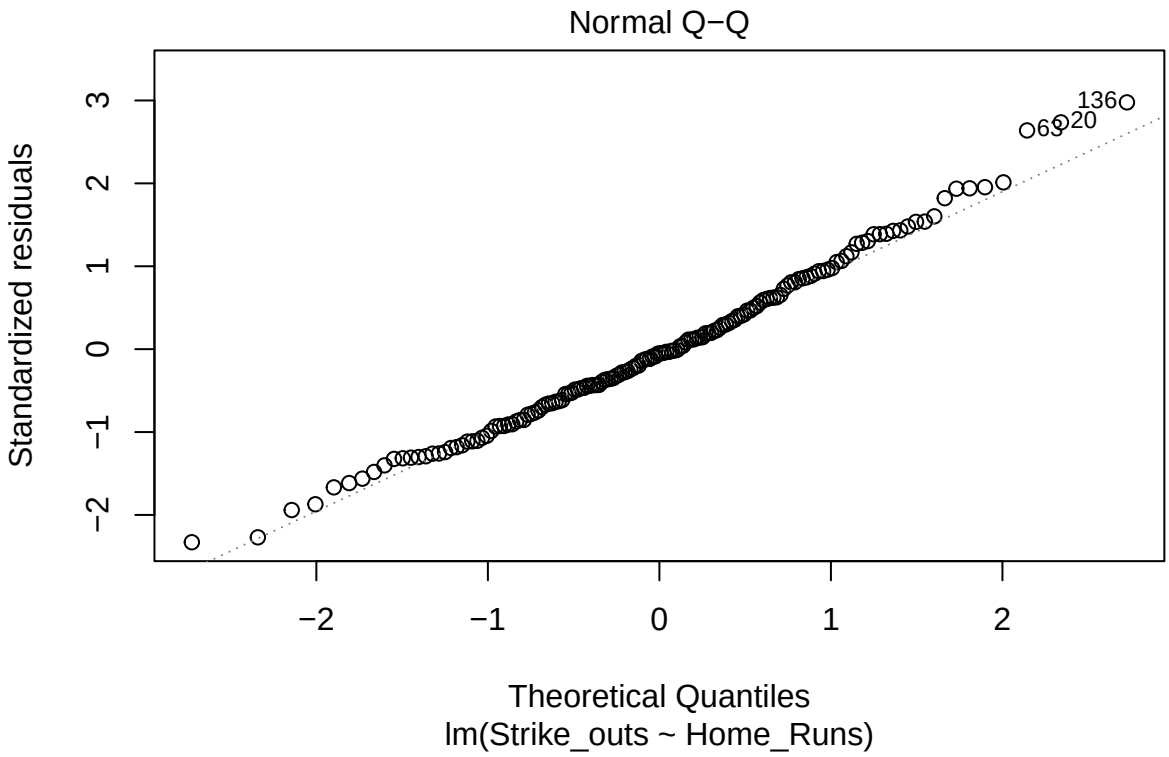


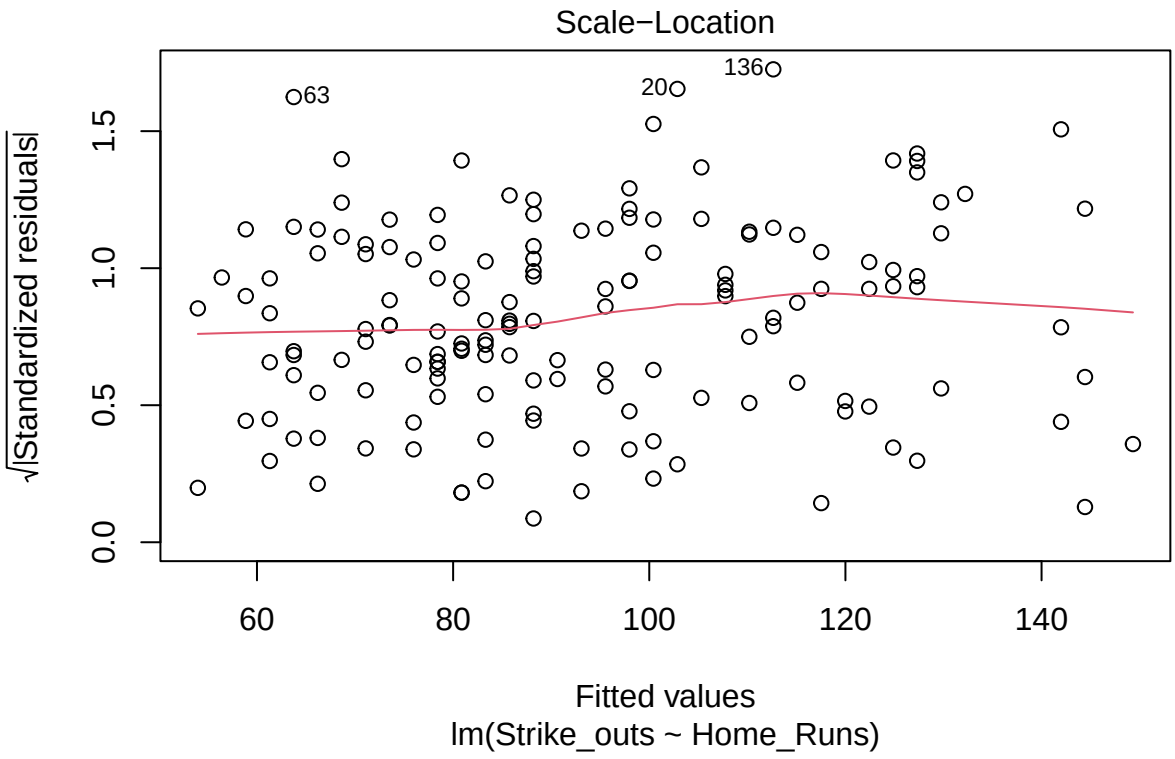

```
Strike_outs <- df2$SO
Home_Runs <- df2$HR

model_SO_HR <- lm(Strike_outs ~ Home_Runs, data = df2)

for(j in 1:3){
  plot(model_SO_HR, which = j)
}
```


Residuals vs Fitted
63
20
136
Residuals
50
0
−50
60
80
100
120
140
Fitted values
lm(Strike_outs ~ Home_Runs)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Strike_outs ~ Home_Runs)
136
20
63
−2
−1
0
1
2
3




## ANOVA

- ANOVA - Analysis of Variance, is another tool we can use to make inferences about our model.
- The linear regression assumptions must hold to justify our inferences about our model, so let's use the `anova()` function on the previous model which satisfied the three assumptions.

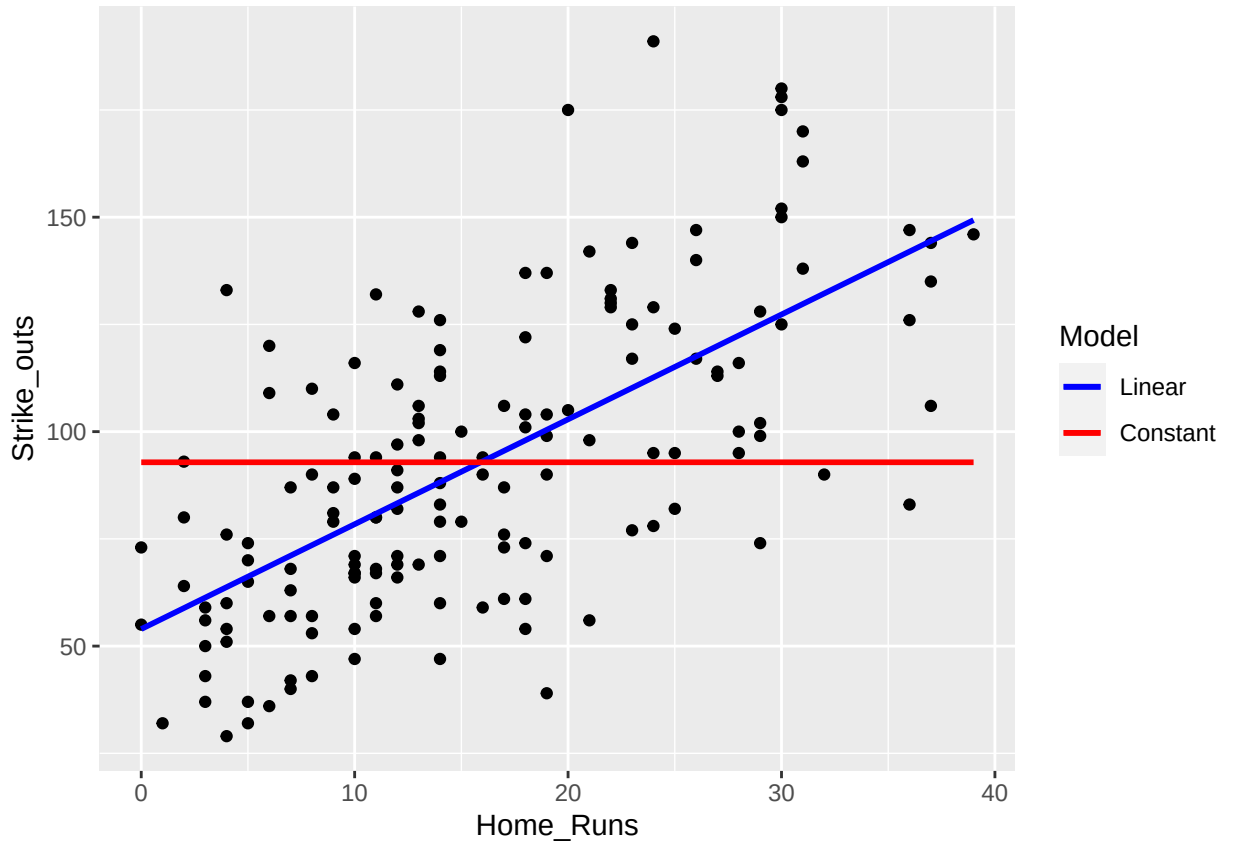
```
anova(model_SO_HR)
```

```
## Analysis of Variance Table
##
## Response: Strike_outs
##             Df Sum Sq Mean Sq F value    Pr(>F)
## Home_Runs    1  81706   81706  116.66 < 2.2e-16 ***
## Residuals 154 107860     700
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The $p$-value basically tells us that `Home_Runs` is a significant predictor of `Strike_outs`. More specifically, it's telling us that our model $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ explains more of the variance in the response compared to the "null model" (intercept only), $Y_i = \beta_0 + \epsilon_i$.

```
df2 %>% ggplot(aes(x = Home_Runs, y = Strike_outs)) +
  geom_point()+
  geom_smooth(aes(color = "Linear"),
              method = "lm",
              formula = "y~x",
              se = F)+
```

```
  geom_smooth(aes(color = "Constant"),
              method = "lm",
              formula = "y~1",
              se = F)+
  scale_color_manual(name = "Model",
                     values = c("Linear" = "blue",
                                "Constant" = "red")
                     )
```




Including Home Runs as a predictor gives us a "better fit" than not including it.

**Questions:** Let's answer the following questions about the `anova()` output:

1) What is the $\hat{\sigma}^2$ value?
2) What is the $R^2$ value?
3) What is our $n$, the number of observations?
4) What is the value for the $t$ statistic for the coefficient $\hat{\beta}_1$
5) What % of variability in strike outs (response variable) is unexplained by a linear relationship with home runs (predictor variable)?

**Answers**

1) $\hat{\sigma}^2 = \text{MSE} = 700.3883158$

```
anova(model_SO_HR)[2,3]
```

```
## [1] 700.3883
```

```
summary(model_SO_HR)$sigma^2
```

```
## [1] 700.3883
```

2) $R^2 = 1 - \frac{RSS}{TSS} = \frac{SS_{reg}}{TSS} = 0.431016$

```
 anova(model_SO_HR)[1,2] /
(anova(model_SO_HR)[1,2] + anova(model_SO_HR)[2,2])
```

```
## [1] 0.431016
```

```
summary(model_SO_HR)$r.squared
```

```
## [1] 0.431016
```

3) The total degrees of freedom (Df) is $n - 1$, so $n = Df_{HR} + Df_{resid} + 1 = 156$.

```
anova(model_SO_HR)[1,1] + anova(model_SO_HR)[2,1] + 1
```

```
## [1] 156
```

4) The $F$-test gives the same result as a pooled two-sample $t$-test where $F$-stat $= t$-stat$^2$, so $t = \sqrt{F} = 10.8008272$.

```
sqrt(anova(model_SO_HR)[1,4])
```

```
## [1] 10.80083
```

```
summary(model_SO_HR)$coef[2,3]
```

```
## [1] 10.80083
```

5) % unexplained variability $= 100(1 - R^2) = 100\frac{RSS}{TSS} = 56.8984012$

```
(anova(model_SO_HR)[2,2] / (anova(model_SO_HR)[1,2] + anova(model_SO_HR)[2,2])) * 100
```

```
## [1] 56.8984
```

```
summary(model_SO_HR)
```

```
##
## Call:
## lm(formula = Strike_outs ~ Home_Runs, data = df2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.422 -17.811  -1.365  16.383  78.354
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  53.9703     4.1772   12.92   <2e-16 ***
## Home_Runs     2.4448     0.2264   10.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.46 on 154 degrees of freedom
## Multiple R-squared:  0.431,  Adjusted R-squared:  0.4273
## F-statistic: 116.7 on 1 and 154 DF,  p-value: < 2.2e-16
```

## Transformations

- See Chapter 7 of Faraway book

*Can we predict At bats from GIDP (Grounded into double plays)?*

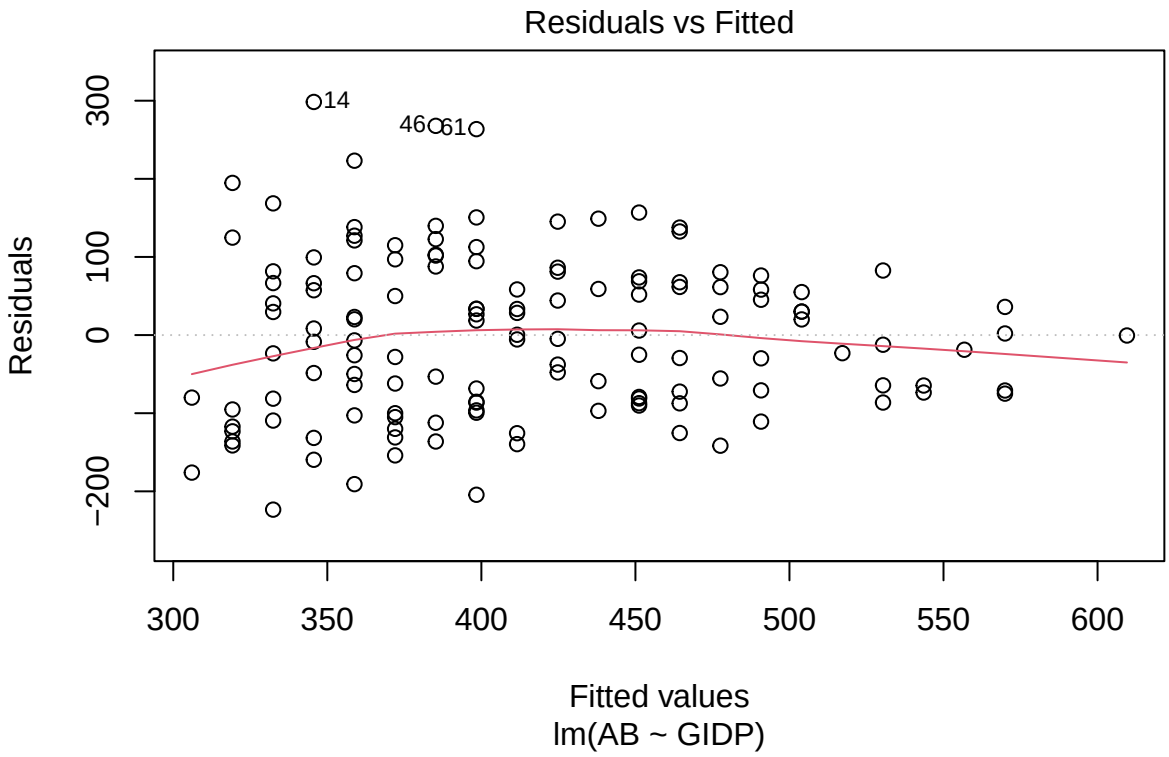
```
df3 <- Batting %>%
        filter(yearID == "2017" & # stats from 2017
        lgID == "NL" & # only from the NL
        G > 70 & # Only players who have played more than 70 games
        SB != 0) # Only players that have at least stolen 1 base.
dim(df3)
```

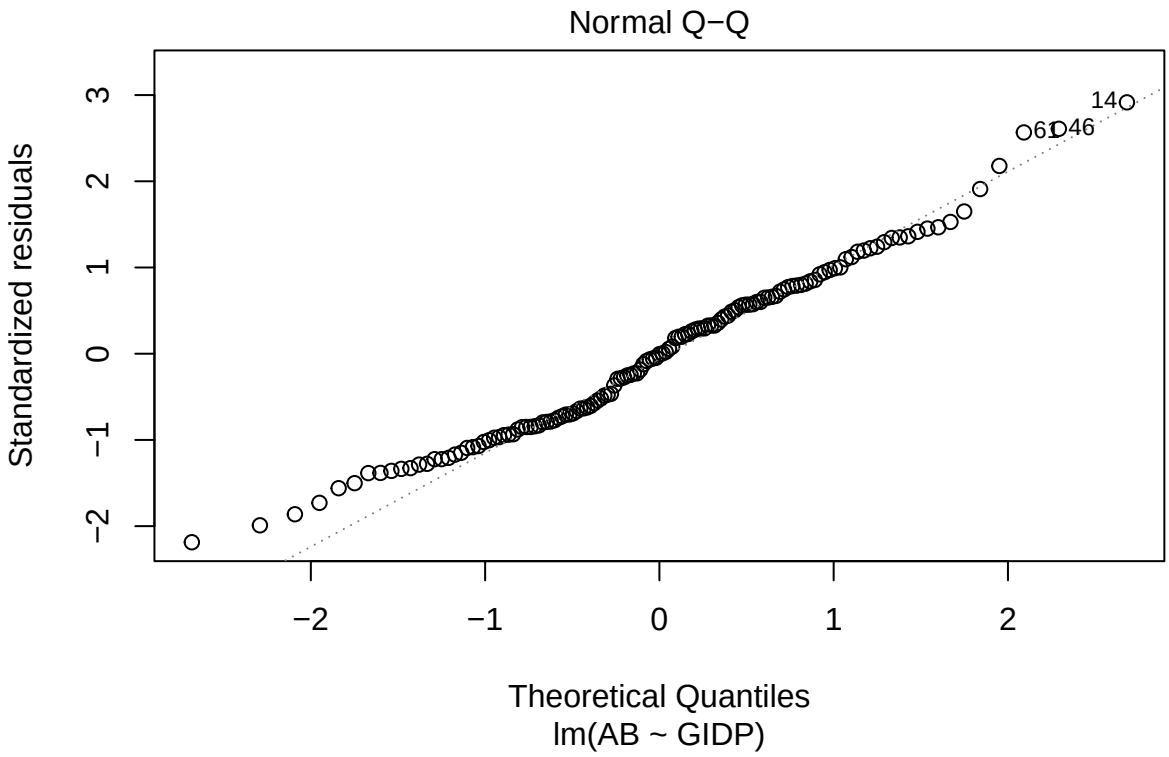
```
## [1] 137  22
```

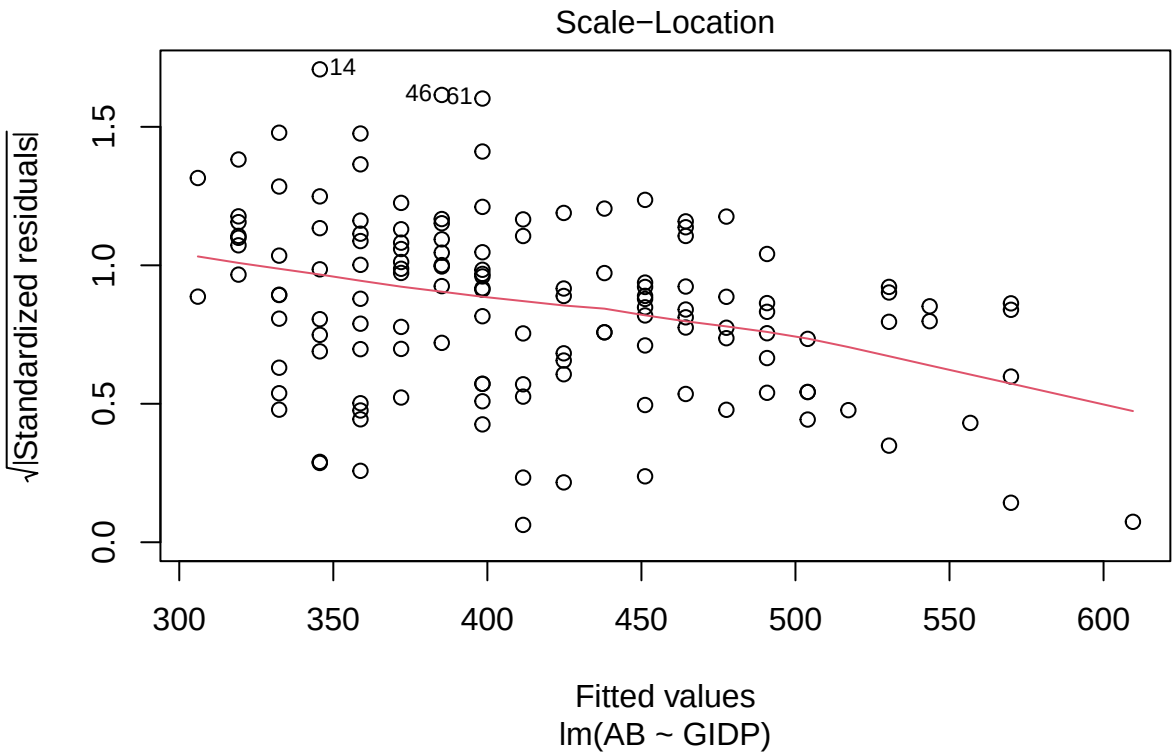
```
model_3 <- lm(AB ~ GIDP, data = df3)
summary(model_3)
```

```
##
## Call:
## lm(formula = AB ~ GIDP, data = df3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -223.420  -81.420   -0.541   68.813  298.383
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  292.832     17.984  16.283  < 2e-16 ***
## GIDP          13.196      1.706   7.734 2.15e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 103.1 on 135 degrees of freedom
## Multiple R-squared:  0.3071, Adjusted R-squared:  0.3019
## F-statistic: 59.82 on 1 and 135 DF,  p-value: 2.15e-12
```

```
for(j in 1:3){
  plot(model_3, which = j)
}
```


Residuals vs Fitted
Residuals
300
100
0
−200
14
46
61
300
350
400
450
500
550
600
Fitted values
lm(AB ~ GIDP)

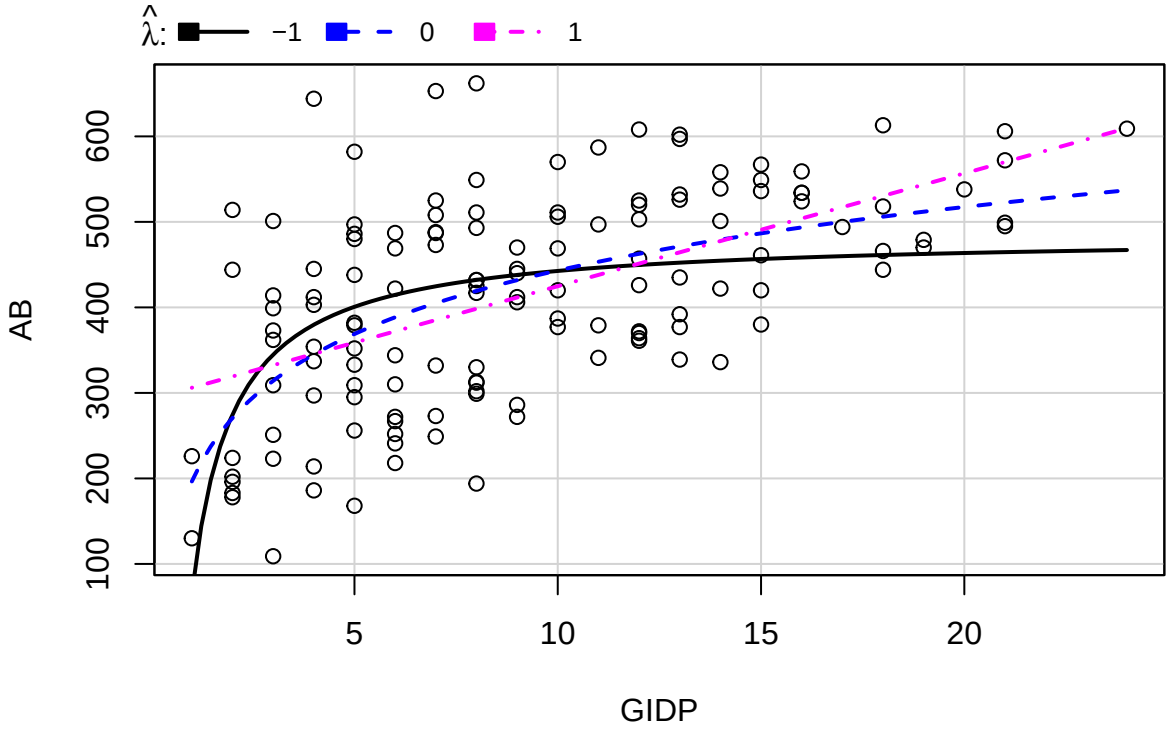

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(AB ~ GIDP)
14
61
46
−2
−1
0
1
2
3




- Possible right skew of residuals (based on two ends of QQ-plot curving up)
- Non-constant variance

We can try transforming our data to fix these issues. We can either transform the response or the predictor, but let's use the `invTranPlot` function from the `car` package to find an appropriate transformation of our predictor.

```
library(car)
invTranPlot(AB ~ GIDP, data = df3, lambda = c(-1, 0, 1), optimal = F)
```




```
##   lambda     RSS
## 1     -1 1533496
## 2      0 1389698
## 3      1 1434801
```

What this function just did was fit 3 models and plotted the results of all 3 of them on the same plot.

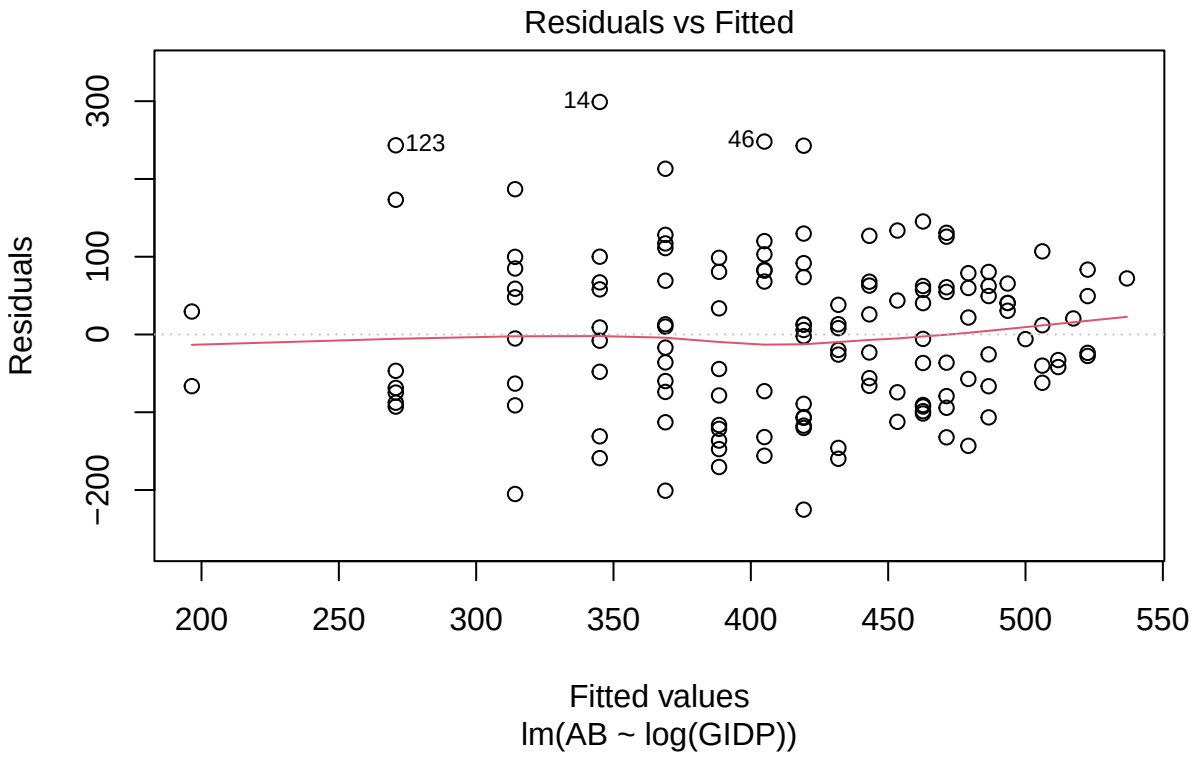
The function transformed our predictor $x$ (`GIDP` in this case) as follows

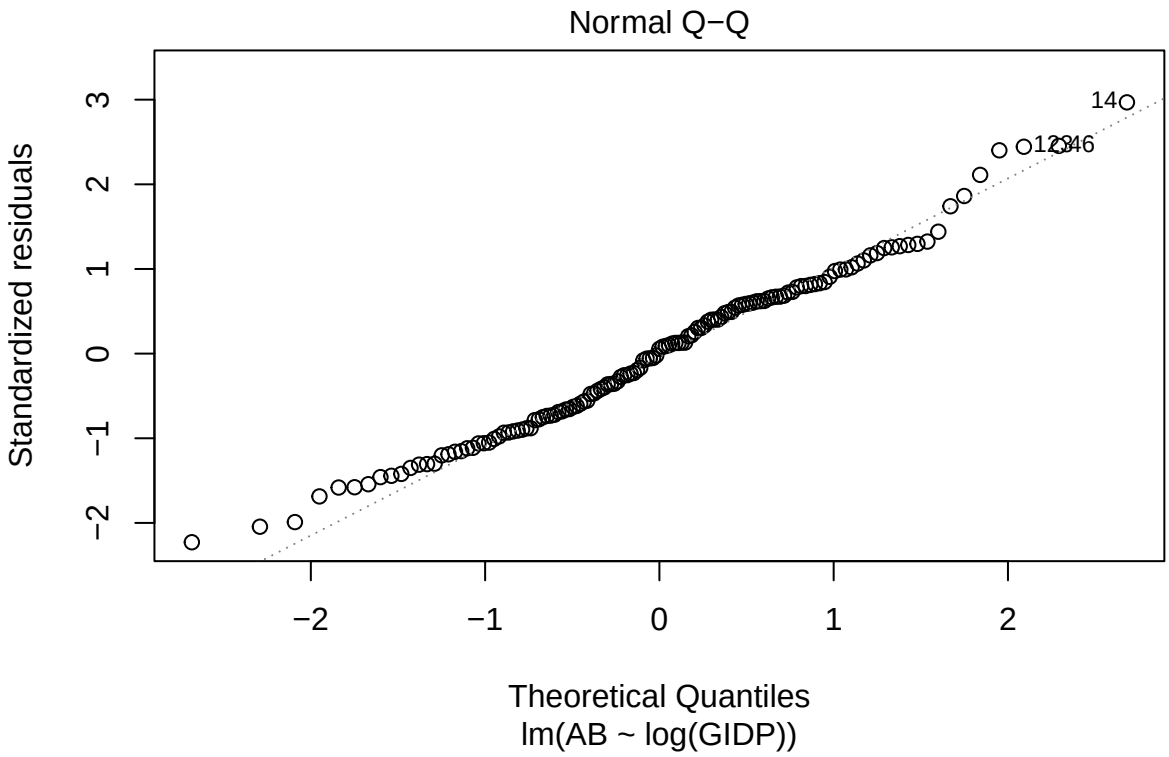
$$v = \begin{cases} \dfrac{x^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(x), & \lambda = 0 \end{cases}.$$

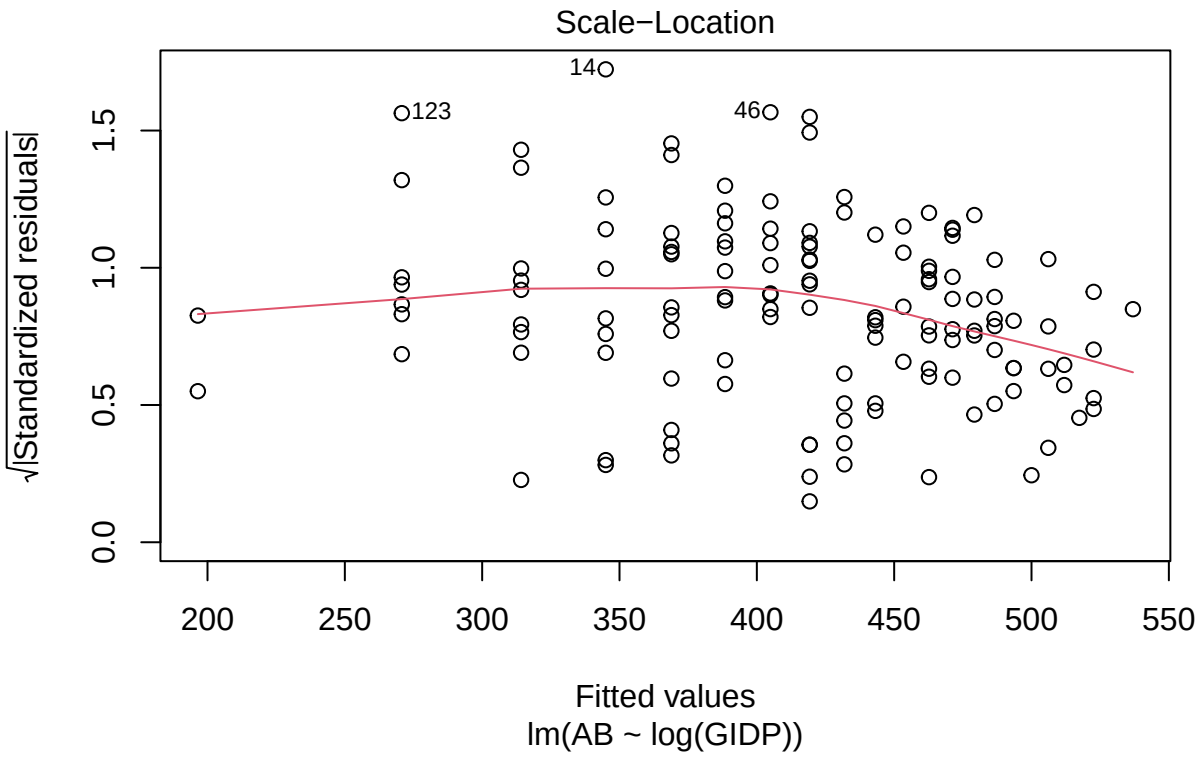
Then a simple linear regression is fit with this new variable $v$ instead. We are (somewhat arbitrarily) choosing to fit these models with $\lambda = -1, 0$, and 1. Notice that $\lambda = 1$ corresponds to no transformation at all.

From the plot above, it looks like $\lambda = 0$ gives the best fit, so we could try moving forward by log transforming `GIDP`. It would be important to re-check our diagnostic plots with this new transformation.

```
for(j in 1:3){
  plot(lm(AB ~ log(GIDP), data = df3), which = j)
}
```


Residuals vs Fitted
Residuals
300
100
0
−200
123
14
46
200
250
300
350
400
450
500
550
Fitted values
lm(AB ~ log(GIDP))


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(AB ~ log(GIDP))
14
12346
−2
−1
0
1
2
3




Not perfect, but better than before.