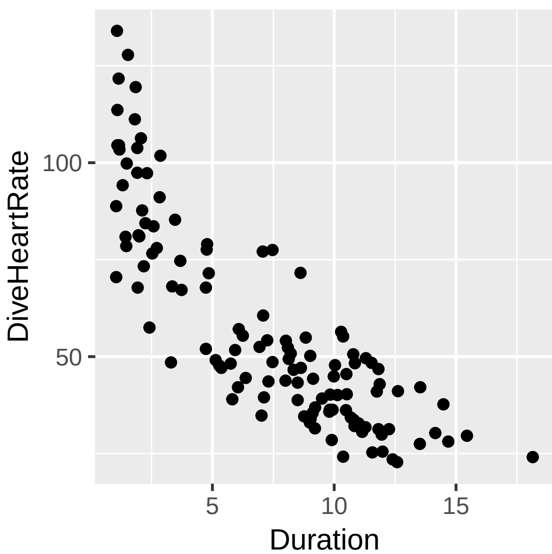
Polynomial Regression

Adapted from De Veaux, Velleman, and Bock

September 07, 2018

Emperor penguins can slow their heart rates while diving. Here's a plot showing 125 observations of penguin dives, with the duration of the penguin's dive on the horizontal axis and the penguin's heart rate on the vertical axis.

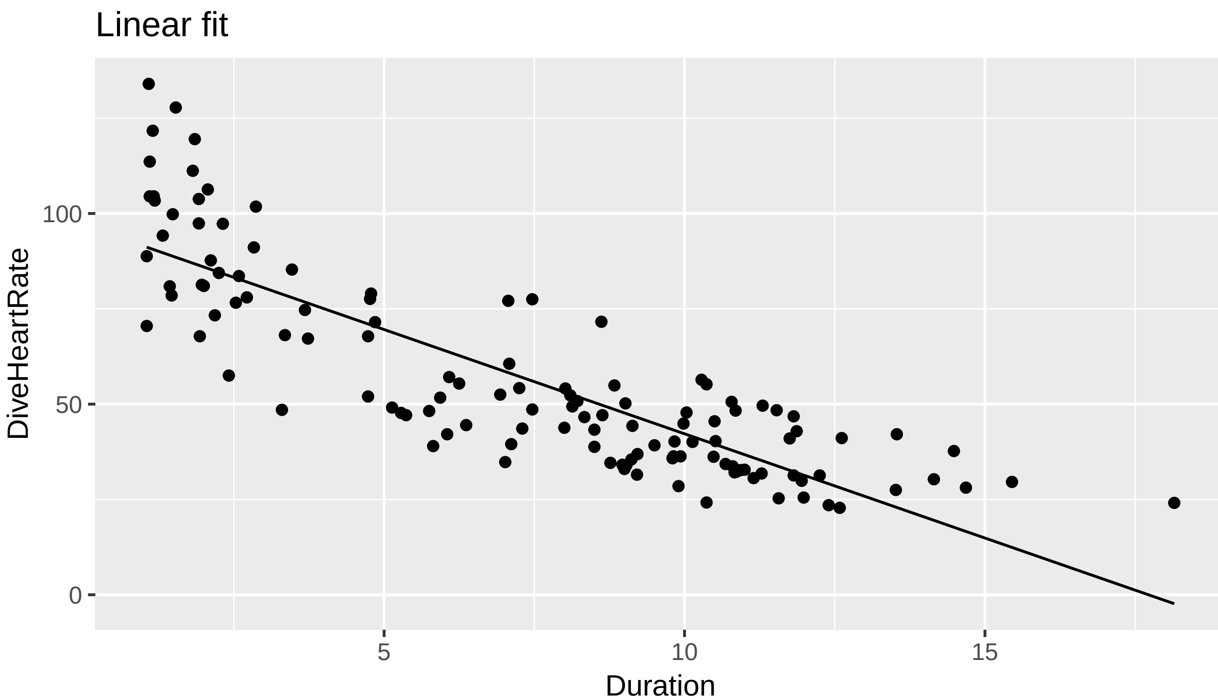
```
library(readr) # for read_csv, which can read csv files from the internet
library(dplyr) # for data manipulation functions
library(ggplot2) # for making plots

penguins <- read_csv("http://www.evanlray.com/data/sdm4/Penguins.csv")

ggplot() +
  geom_point(data = penguins, mapping = aes(x = Duration, y = DiveHeartRate))
```



Linear Fit

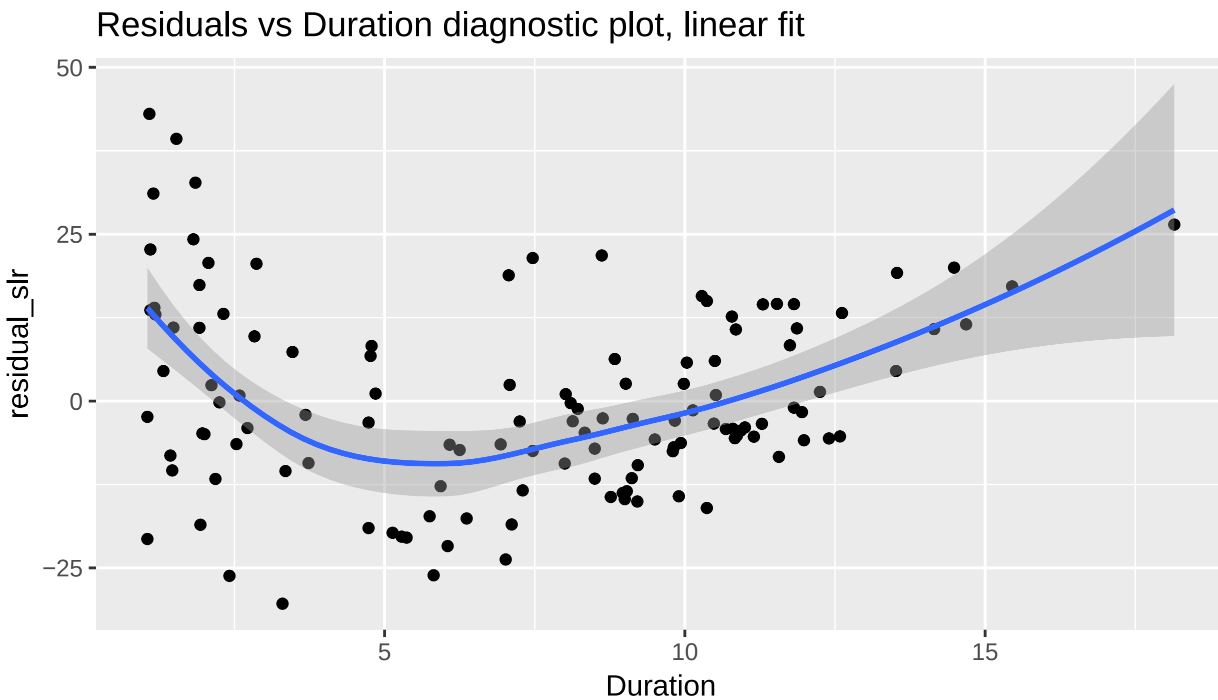
Is a simple linear regression model good enough? Let's fit a model and look at some diagnostic plots to find out:

```
slr_fit <- lm(DiveHeartRate ~ Duration, data = penguins)
predict_slr <- function(x) {
  predict(slr_fit, data.frame(Duration = x))
}

ggplot(data = penguins, mapping = aes(x = Duration, y = DiveHeartRate)) +
  geom_point() +
  stat_function(fun = predict_slr) +
  ggtitle("Linear fit")
```




```
penguins <- penguins %>%
  mutate(
    residual_slr = residuals(slr_fit)
  )

ggplot(data = penguins, mapping = aes(x = Duration, y = residual_slr)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Residuals vs Duration diagnostic plot, linear fit")
```

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



There is a clear trend in the residuals. Let's try fitting a parabola instead.

Quadratic Fit

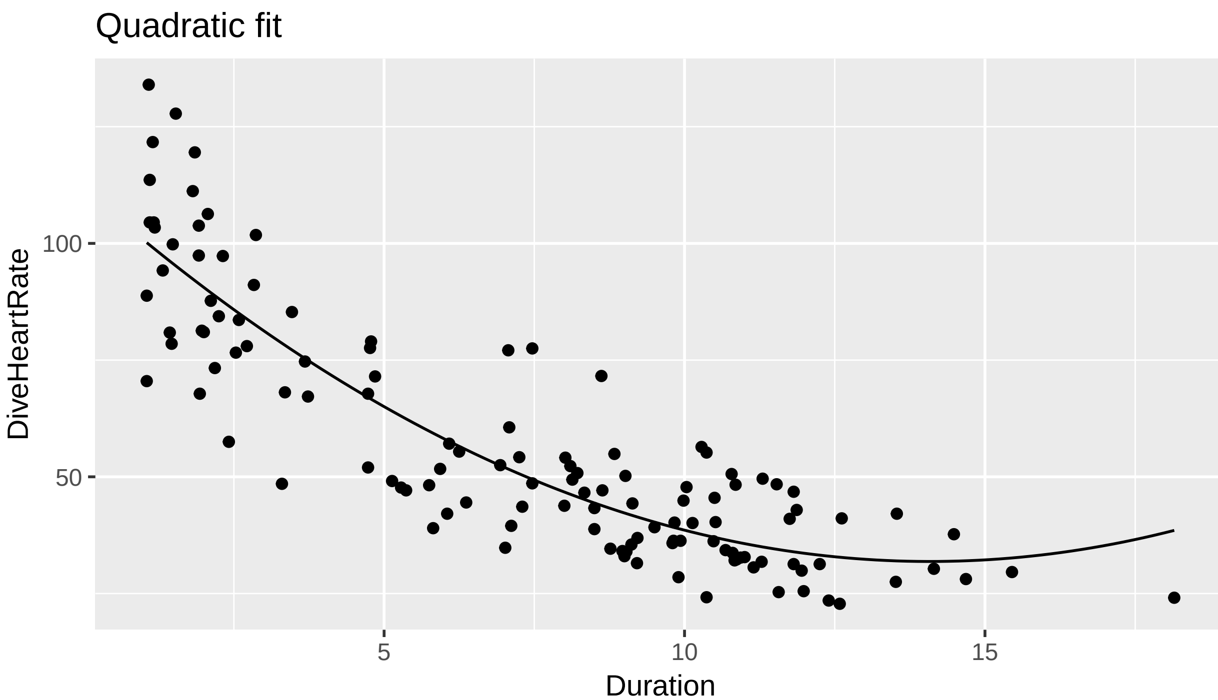
Note the addition of `+ I(Duration^2)` in the model formula.

```
quad_fit <- lm(DiveHeartRate ~ Duration + I(Duration^2), data = penguins)
summary(quad_fit)
```

```
##
## Call:
## lm(formula = DiveHeartRate ~ Duration + I(Duration^2), data = penguins)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.115  -8.289  -1.567   8.016  34.187
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   111.60991    3.32024  33.615  < 2e-16 ***
## Duration      -11.32555    0.99734 -11.356  < 2e-16 ***
## I(Duration^2)   0.40212    0.06585   6.107 1.25e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.4 on 122 degrees of freedom
## Multiple R-squared:  0.782,  Adjusted R-squared:  0.7784
## F-statistic: 218.8 on 2 and 122 DF,  p-value: < 2.2e-16
```

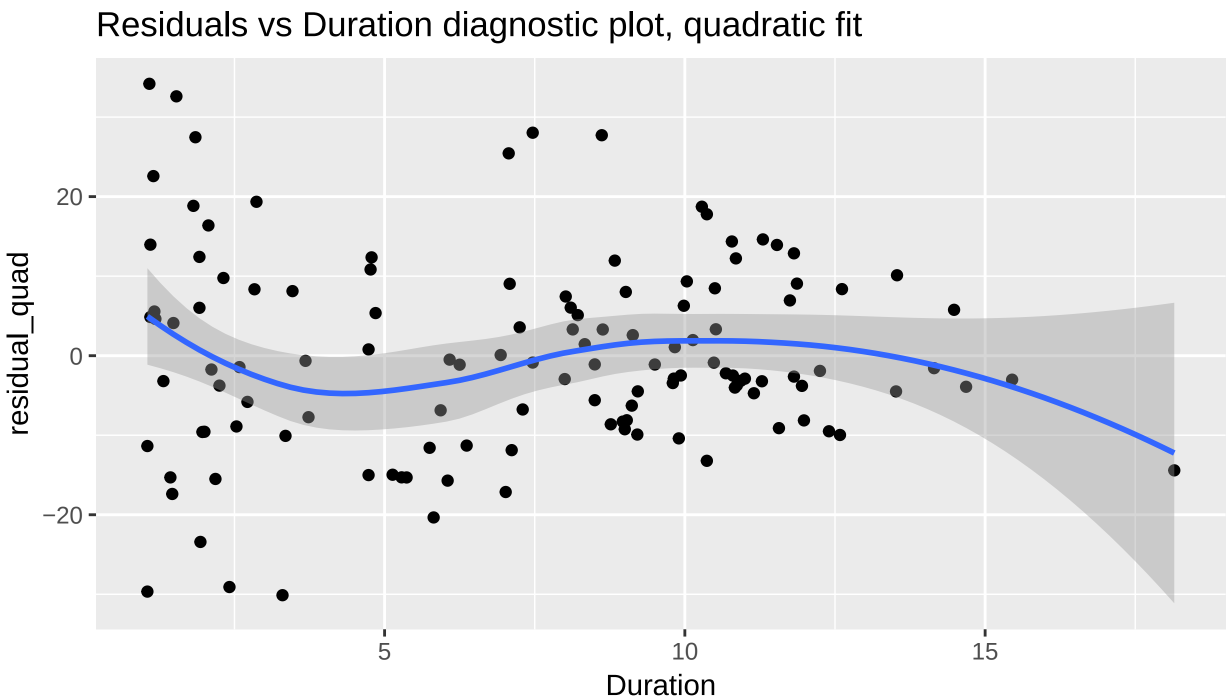
```
predict_quad <- function(x) {
  predict(quad_fit, data.frame(Duration = x))
}
```

```r
ggplot(data = penguins, mapping = aes(x = Duration, y = DiveHeartRate)) +
  geom_point() +
  stat_function(fun = predict_quad) +
  ggtitle("Quadratic fit")
```




```r
penguins <- penguins %>%
  mutate(
    residual_quad = residuals(quad_fit)
  )

ggplot(data = penguins, mapping = aes(x = Duration, y = residual_quad)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Residuals vs Duration diagnostic plot, quadratic fit")
```

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



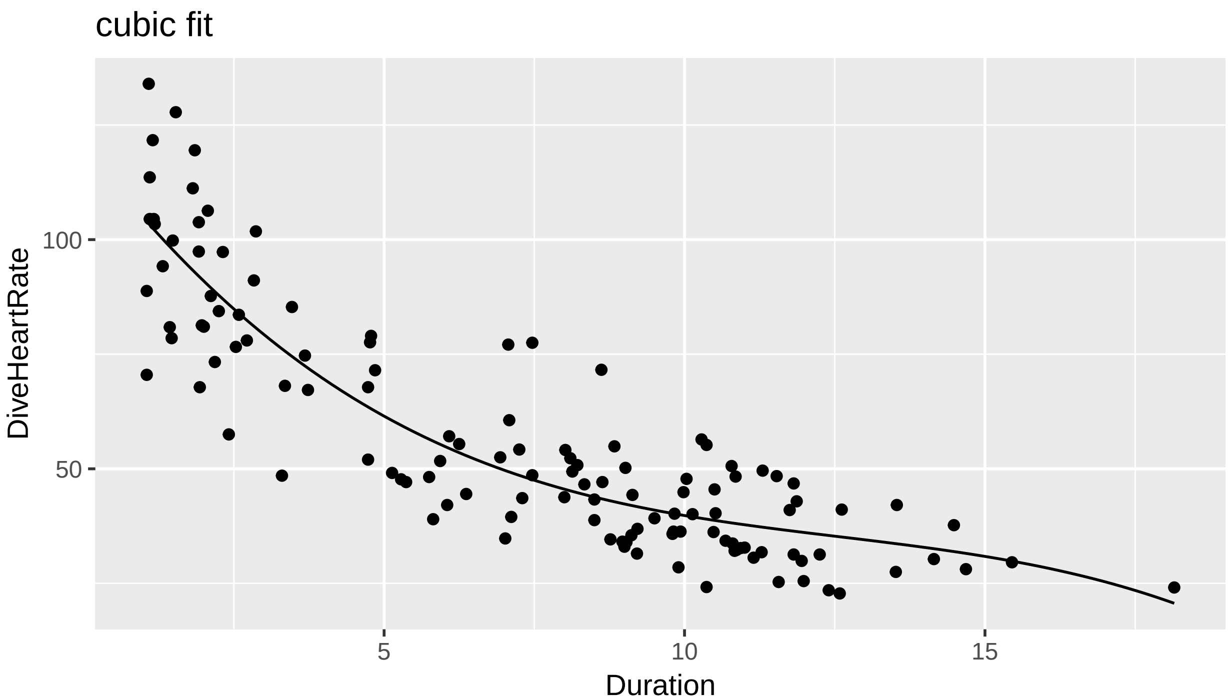
Not as much of a trend... What happens if we fit a cubic polynomial?

Cubic Fit

```
cubic_fit <- lm(DiveHeartRate ~ Duration + I(Duration^2) + I(Duration^3), data = penguins)
summary(cubic_fit)
```

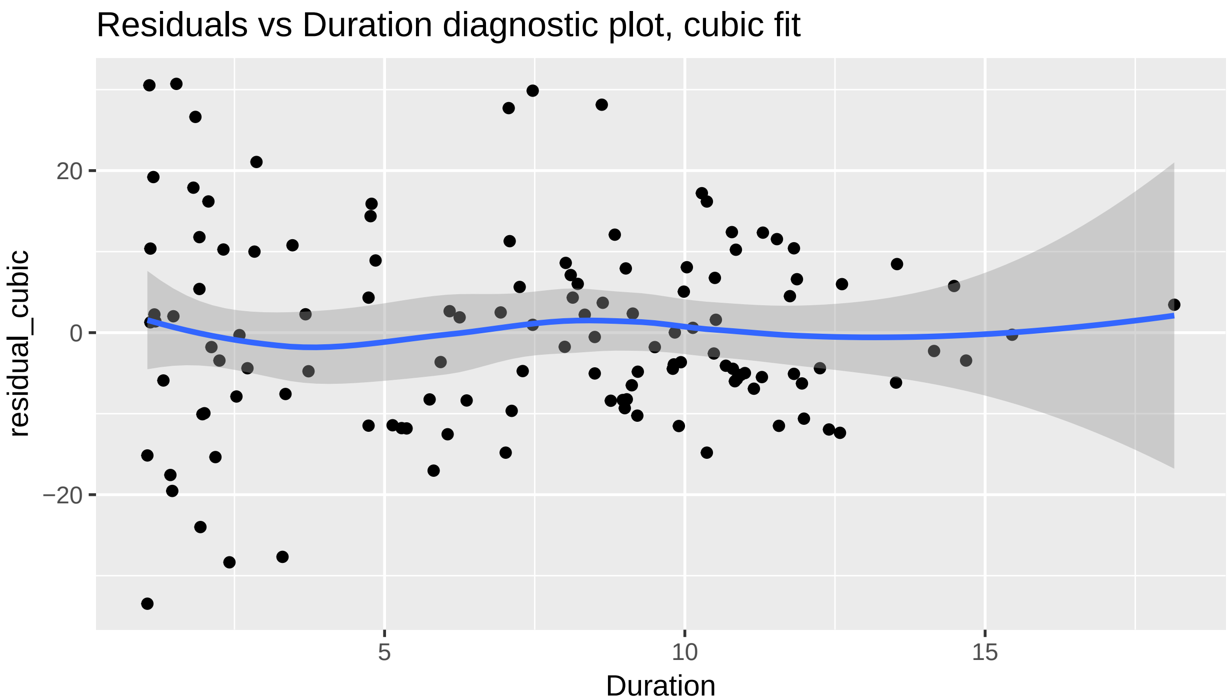
```
##
## Call:
## lm(formula = DiveHeartRate ~ Duration + I(Duration^2) + I(Duration^3),
##     data = penguins)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.458  -7.882  -1.752   7.109  30.710
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   120.74815    4.97143  24.288  < 2e-16 ***
## Duration      -17.26431    2.63037  -6.563 1.38e-09 ***
## I(Duration^2)   1.24772    0.35363   3.528 0.000592 ***
## I(Duration^3)  -0.03308    0.01360  -2.432 0.016478 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.16 on 121 degrees of freedom
## Multiple R-squared:  0.7921, Adjusted R-squared:  0.787
## F-statistic: 153.7 on 3 and 121 DF,  p-value: < 2.2e-16
```

```
predict_cubic <- function(x) {
  predict(cubic_fit, data.frame(Duration = x))
}

ggplot(data = penguins, mapping = aes(x = Duration, y = DiveHeartRate)) +
  geom_point() +
  stat_function(fun = predict_cubic) +
  ggtitle("cubic fit")
```




```
penguins <- penguins %>%
  mutate(
    residual_cubic = residuals(cubic_fit)
  )

ggplot(data = penguins, mapping = aes(x = Duration, y = residual_cubic)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Residuals vs Duration diagnostic plot, cubic fit")
```

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



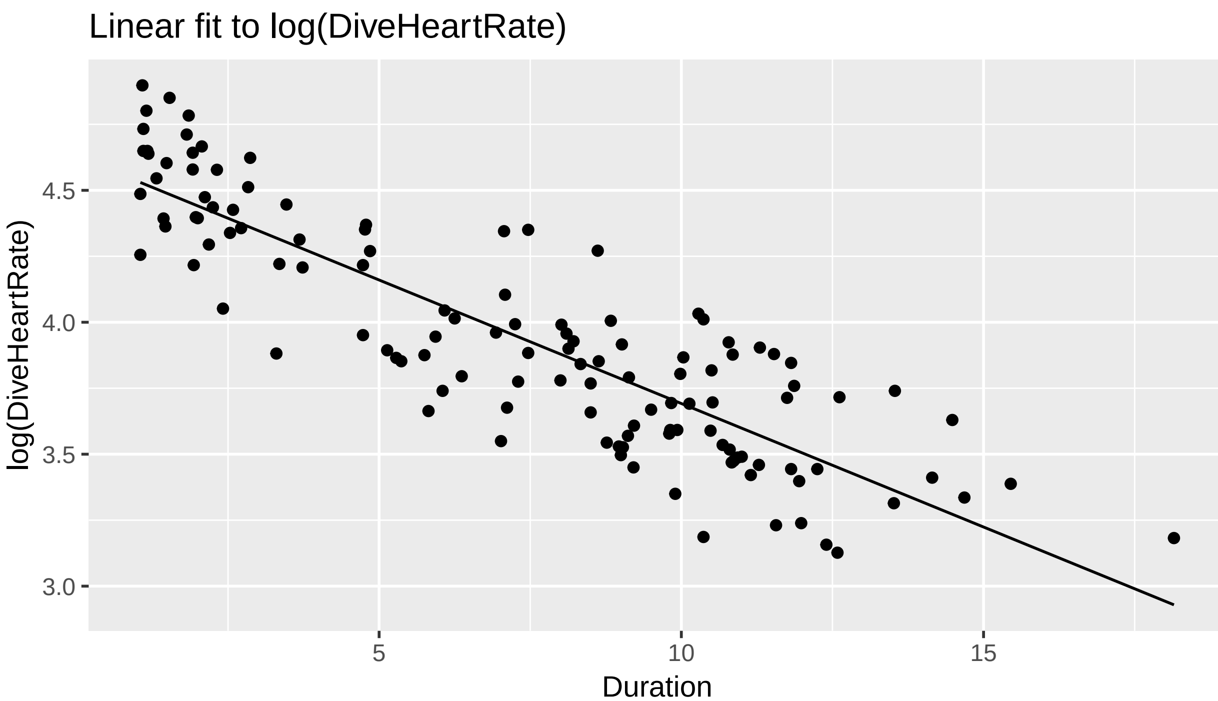
We can also get the same model fit another way, using `poly()` instead of `I()`:

```
cubic_fit <- lm(DiveHeartRate ~ poly(Duration, 3), data = penguins)
summary(cubic_fit)
```

```
##
## Call:
## lm(formula = DiveHeartRate ~ poly(Duration, 3), data = penguins)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.458  -7.882  -1.752   7.109  30.710
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          56.924      1.088  52.343  < 2e-16 ***
## poly(Duration, 3)1 -248.097     12.159 -20.405  < 2e-16 ***
## poly(Duration, 3)2   75.734     12.159   6.229 7.07e-09 ***
## poly(Duration, 3)3  -29.571     12.159  -2.432   0.0165 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.16 on 121 degrees of freedom
## Multiple R-squared:  0.7921, Adjusted R-squared:  0.787
## F-statistic: 153.7 on 3 and 121 DF,  p-value: < 2.2e-16
```

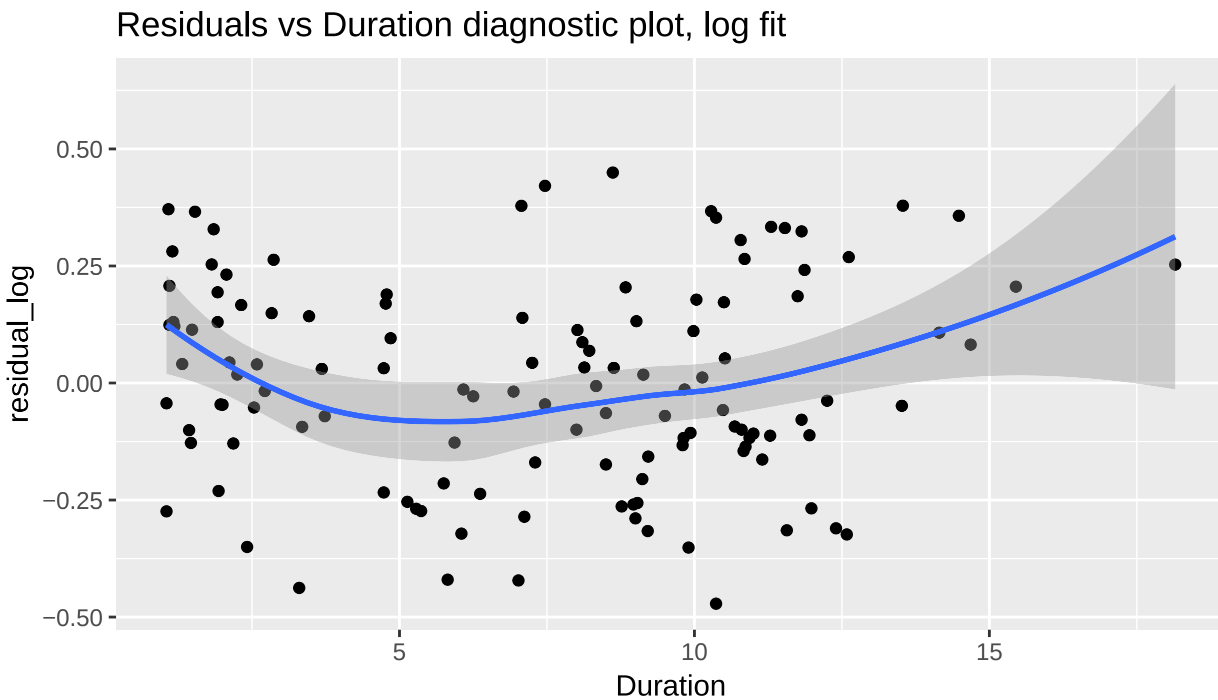
Another Approach... Data Transformation

```
log_fit <- lm(log(DiveHeartRate) ~ Duration, data = penguins)
predict_log <- function(x) {
  predict(log_fit, data.frame(Duration = x))
}

ggplot(data = penguins, mapping = aes(x = Duration, y = log(DiveHeartRate))) +
  geom_point() +
  stat_function(fun = predict_log) +
  ggtitle("Linear fit to log(DiveHeartRate)")
```




```
penguins <- penguins %>%
  mutate(
    residual_log = residuals(log_fit)
  )

ggplot(data = penguins, mapping = aes(x = Duration, y = residual_log)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Residuals vs Duration diagnostic plot, log fit")
```

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


Residuals vs Duration diagnostic plot, log fit
0.50
0.25
0.00
−0.25
−0.50
residual_log
5
10
15
Duration