Linear Regression

Nicholas Kortessis

Linear Regression

The functional idea with linear regression is that the character of an individual can be partly predicted with information about another character of that same individual. Finding this information is called "regression" and linear regression is a special case where one character (called a response) can be predicted by a linear function of some other character (called a predictor).

Typically, we write the predictor with symbol X and we write the response with the symbol Y. And when we pay attention to a single individual i, the predictor for individual i is X_i and the response for individual i is Y_i.

A typical example is that you want to might think that there is information about the length of different parts of a flower might be related to each other. R has data on the length of different parts of flowers for different species of plants in the genus *Iris*. This data were oridianlly collected by Edgar Anderson and published in 1935. If you are interested in the original paper, the citation is

- Anderson, Edgar. 1935. The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, **59**, 2–5.

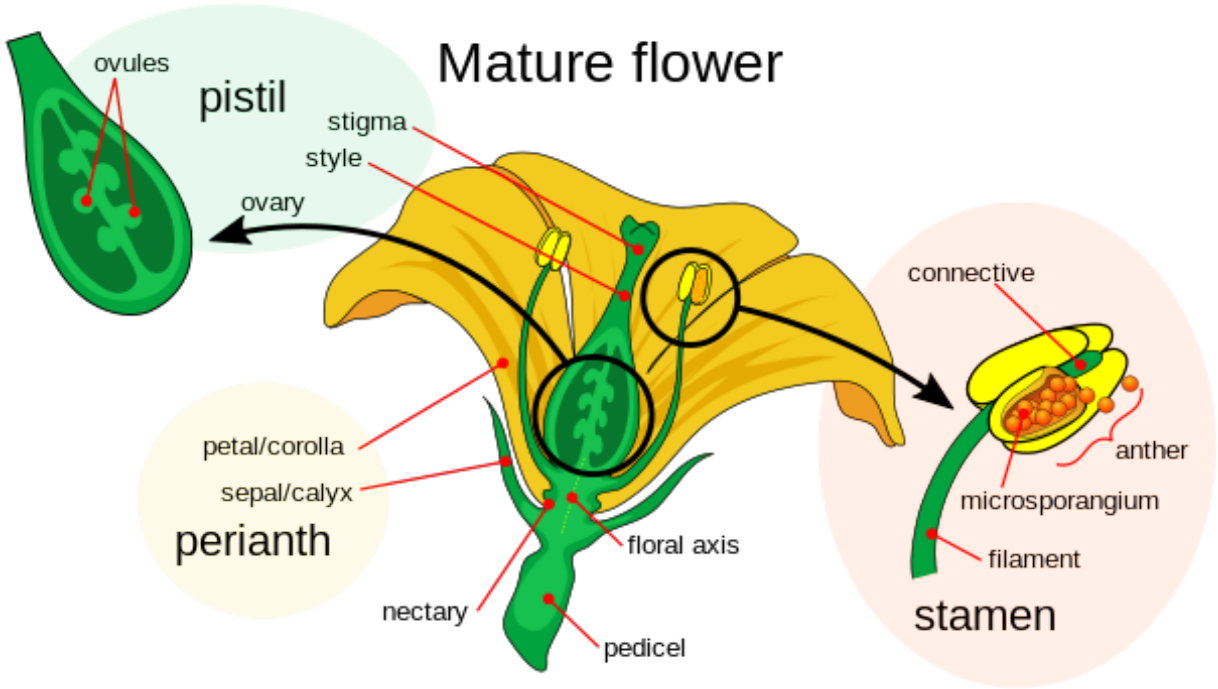
```
data("iris")
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

This data set has 50 plants of three species and measurements of sepal and petal width and height for each individual. (Remember, petals are the colorful leaf type structures on flowers and sepals are typically green leaf type structures at the very base of a flower.)



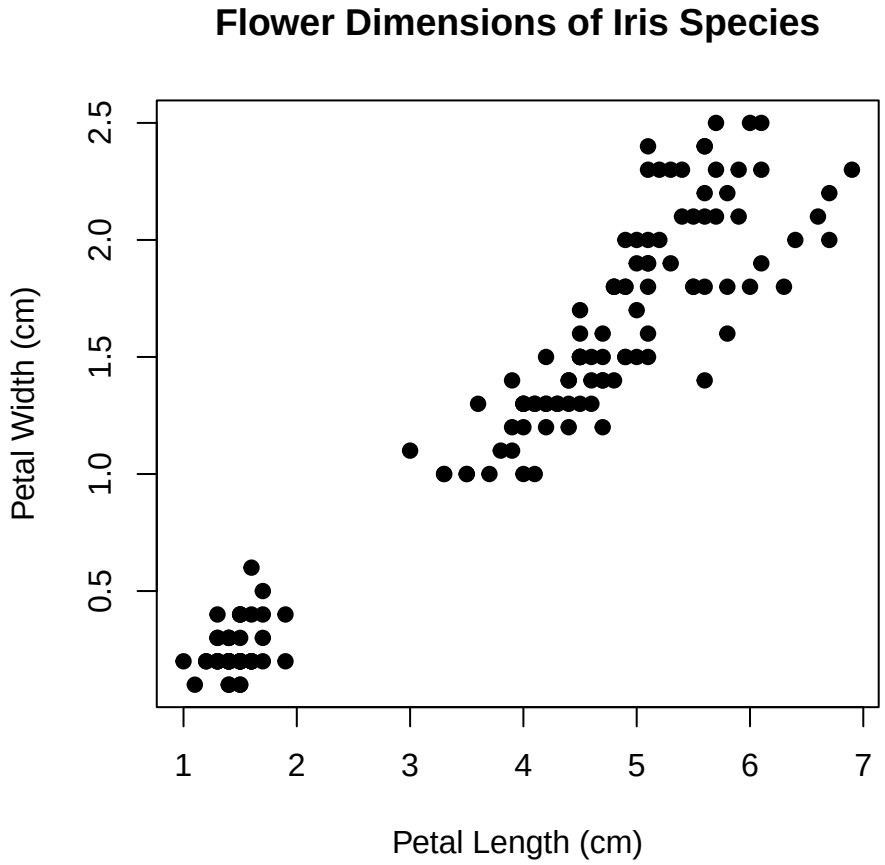
The characteristics of individual 4 in the data set are

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 4          4.6         3.1          1.5         0.2  setosa
```

This is a natural dataset to ask some questions about the **relationship** between multiple characteristics in individuals.

1. Do plants with longer petals have wider petals?
2. If so, how much does wider are petals for a given length?
3. How much can one understand about petal width from length?
4. If I know petal length, what should I expect petal width to be?

Each of these questions can be answered with regression. To make things simple, let's start with just a plot that helps us think about all these questions. It's just a plot of the petal length and width of each individual. Each point represents an individual.



It definitely looks like longer petals are also wider (and vice versa). It also looks like we can say that the increase in petal width with petal length is approximately linear, meaning we can draw a straight line through it.

We can write this mathematically. Let Y_i be the petal width of individual i and let X_i be the petal length of individual i. We are going to create a *probability model* to describe the relationship between Y and X as follows

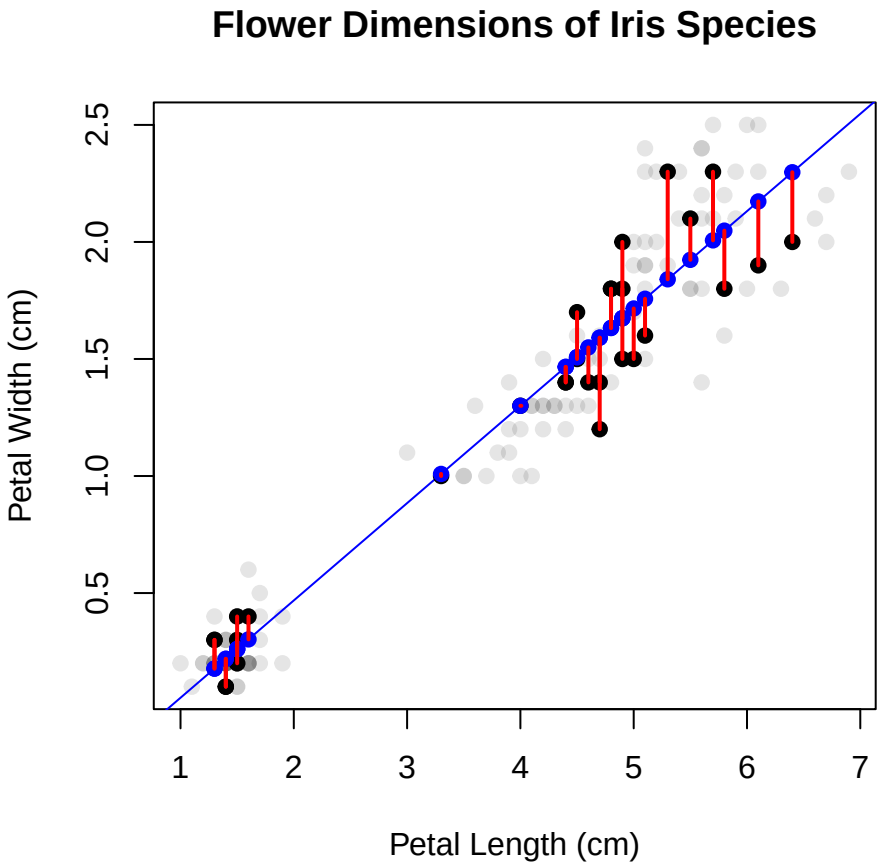
$$Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{a line}} + \underbrace{\epsilon_i}_{\text{residual}}$$

This model has two components: a line and a residual.

The **line** represents, well, any line that we could draw through the data points. The line has two parameters to describe it, an intercept β_0 and a slope, β_1. The intercept represents the y-value of the line when $X = 0$, and the slope represents how much the y-value of the line changes with one unit increase in the value of X.

The **residual** represents all that is leftover about the value of Y_i that is NOT described by the line. In this way, the residuals are very much like the errors in a linear model.

The figure below shows the line and the residual for a couple of points in the *Iris* dataset.

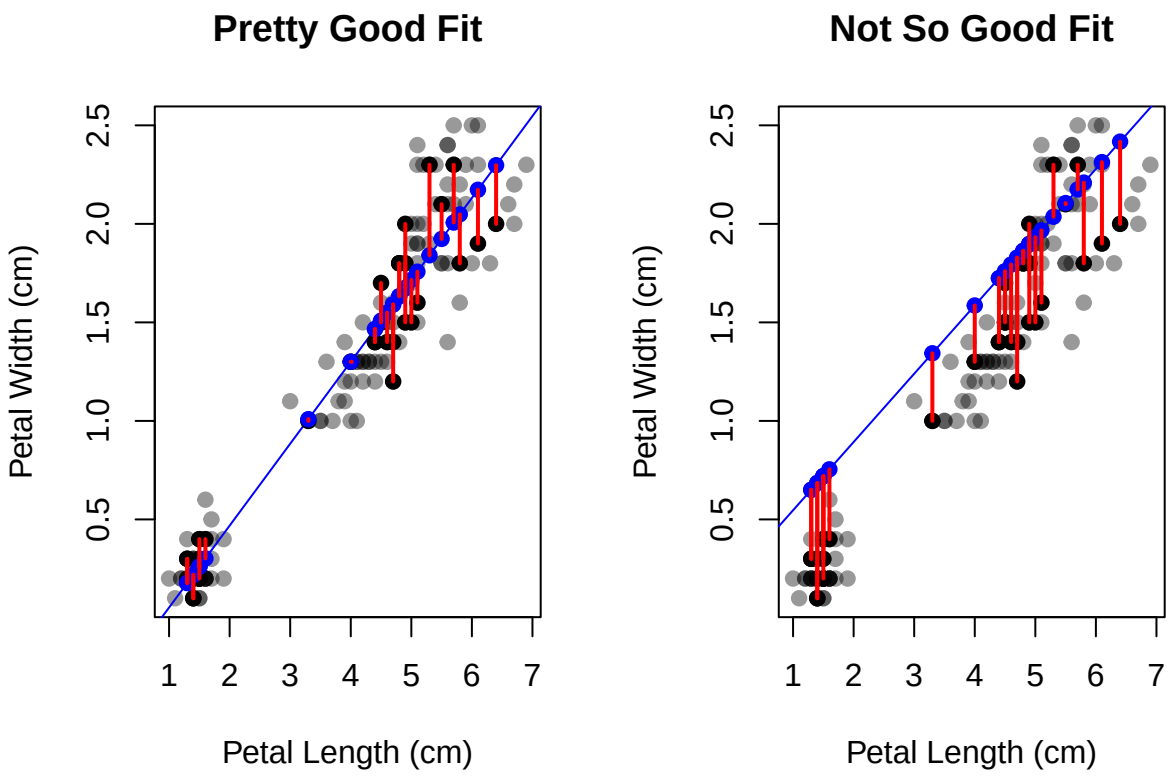


In this figure, the full line is given in blue, and the particular values of the line for specific individuals is given by blue dots. The residuals are given by the red lines that move from the blue line to the actual data points.

This line is actually what is considered to be the ***best fit line***. But how does one determine a line that fits the best?

Finding the best fit line

To find a best fit line, one needs to first describe a method for describing fit. To illustrate fit and lack of fit, look at the figure below with two different lines.



Visually, you can see that the panel on the right doesn't fit as well. You might also notice that this is because the blue points given by the line are consistently further away from the actual data points. The distance between the blue points given by the line and the actual data values is exactly the residuals, given by the red lines. Mathematically, we just ask for the difference between the data points and the line, which is

$$Y_i - (\beta_0 + \beta_1 X_i) = \beta_0 + \beta_1 X_i + \epsilon_i - (\beta_0 + \beta_1 X_i) = \epsilon_i,$$

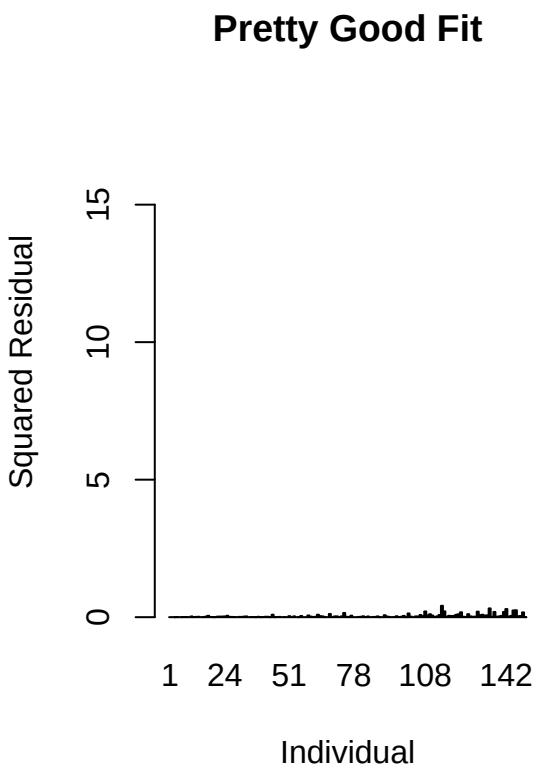
which is just the residuals!

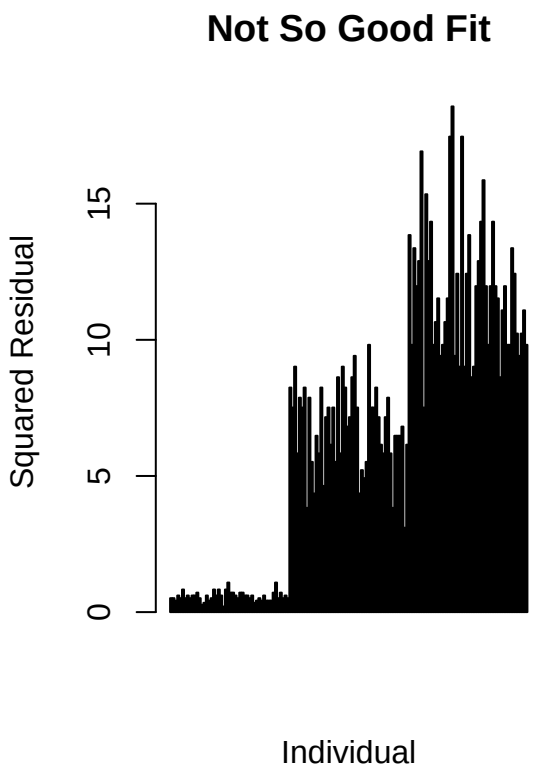
Poor fit is then given by larger distances of residuals. The more the residual distance of the points from any given line, the worse they fit.

We have many times before measured how much individuals differ from some specified value whenever we are calculating variability. We measure variability using squares. Here, we can measure how much the line differs from the data points by looking at **squared residuals,** ϵ^2.

If we calculate the squared residuals for each data point in each of these panels, we get the following

```
par(mfcol = c(1,2))
barplot(mdl$residuals^2,
     xlab = 'Individual', ylab = 'Squared Residual',
     main = 'Pretty Good Fit',
     ylim = c(0, max(residuals^2)))

barplot(residuals^2, xlab = 'Individual', ylab = 'Squared Residual',
        main = 'Not So Good Fit')
```





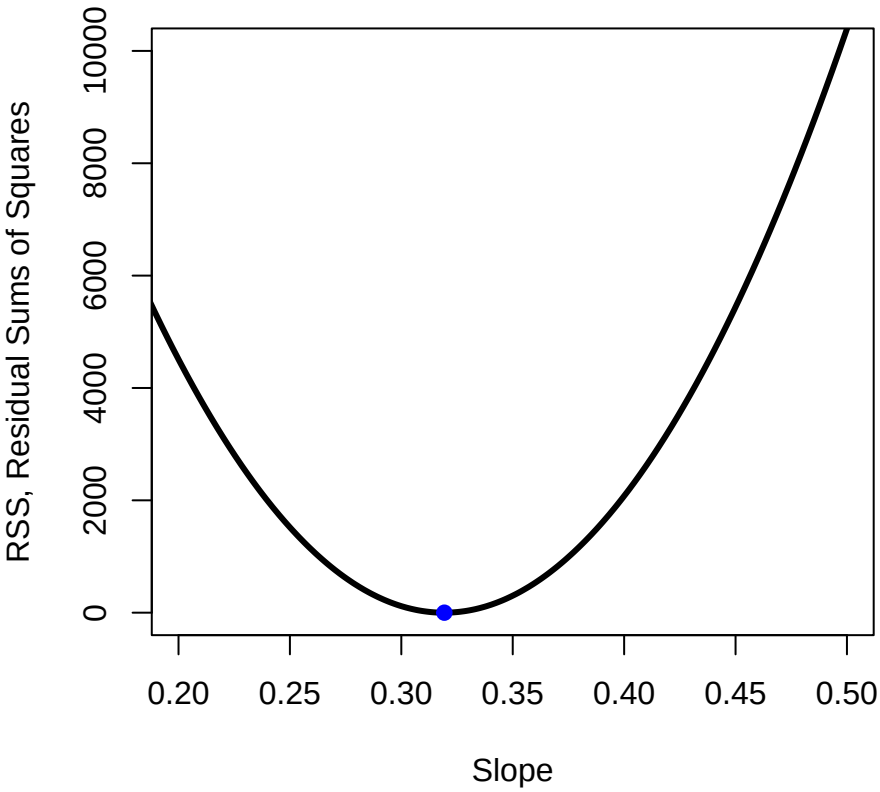
Because the first line fits so good, the squared residuals are barely perceptible on the graph. And for the poor fitting line, the squared residuals look much larger. One way to evaluate lack of fit is by looking at the total of these squared residuals

$$\text{Residual Sums of Squares} = RSS = \sum_{i=1}^{n} \epsilon^2.$$

Lines fit worse when this residual sums of squares is higher and lines fit better when the residual sums of squares is smaller. How does one find the best fit line then? You simply find the line with the smallest residual sums of squares.

What this means in practice is to find the values of β_0 and β_1 that leads to the smallest value of RSS. This can be done using the tools of calculus, but here is a way to think about how this might work.

Let's pick $\beta_0 = 0$ because this says that flower petals have zero width when their flower petals have zero length. Seems pretty reasonable. Now we can scan across a bunch of values of β_1, the slope and calculate RSS each time. When we do that, we get the following plot.



The blue points is the minimum value of the residual sums of squares. The slope at this point is $\hat{\beta}_1 = 0.32$. We give this parameter a "hat" to signify that it is an **estimate** of the slope. This estimate is found by **minimizing residual sums of squares.** This particular estimate is therefore known as the **least squares estimate** (LSE) or sometimes known as **ordinary least squares** (OLS) estimation.

Luckily, we don't need to do this process every time. Using the tools of calculus, we have the following equation for the LSE of the slope and the intercept.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

and

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}.$$

What this says is that you can estimate the slope using the data points and then use that to estimate the intercept. (Remember from calculus that if you want to find the minimum of a function $f(x)$, you find the x-values where the slope of the function is zero, $df(x)/dx = 0$, and the curvature of the function is positive, $d^2f(x)/dx^2 > 0$. You can see from the figure that the slope of the line at the blue point is zero and the curvature is positive).

If we use this data on flower petals and widths in the equations for the least squares estimates, we find that the least squares estimates for the slope and intercept are

$$\hat{\beta}_1 = 0.416$$

and

$$\hat{\beta}_0 = -0.363.$$

These are subtly different than the method used above because you actually need to find the **combination of** slopes and intercepts that **together minimize RSS**. We didn't exactly do that because we just picked a value of the intercept of zero and then found the slope that minimized RSS.

Interpreting the slope estimator, $\hat{\beta}_1$

The equations for the slopes estimator looks intimidating, but it has a very nice interpretation, if you just give yourself a chance to figure out what is going on.

To see what is going, first consider the numerator. The denominator is always positive, so it doesn't matter as much at this point.

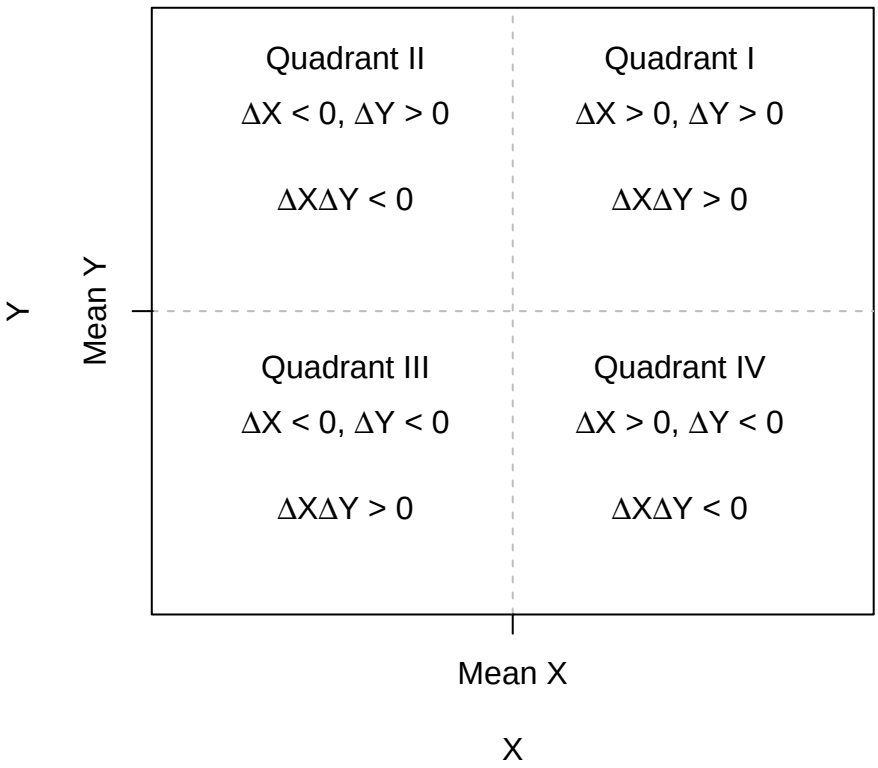
Here is the numerator of $\hat{\beta}_1$:

$$\sum_{i=1}^{n} \overbrace{(X_i - \bar{X})}^{\Delta X} \overbrace{(Y_i - \bar{Y})}^{\Delta Y}.$$

I've labelled these as ΔX and ΔY. These values are how much a data point's x and y values differ from the average. When the point is above average, it's delta value will be positive. When the point is below average, the delta value will be negative. Importantly, this is a product of two delta values, and so we have the following possibilities for each point:

	Below Average X ($\Delta X < 0$)	Above Average X ($\Delta X > 0$)
Above Average Y ($\Delta Y > 0$)	$\Delta X \Delta Y < 0$ (quadrant II)	$\Delta X \Delta Y > 0$ (quadrant I)
Below Average Y ($\Delta Y < 0$)	$\Delta X \Delta Y > 0$ (quadrant III)	$\Delta X \Delta Y < 0$ (quadrant IV)

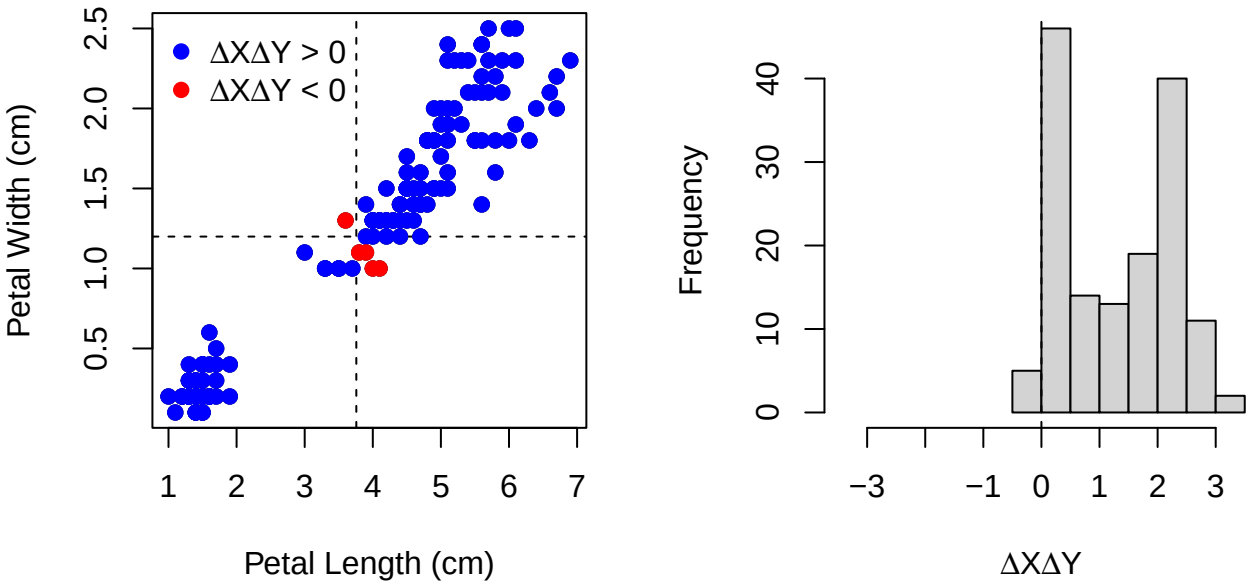
The graph below shows the quadrants and what the sign of $(X_i - \bar{X})(Y_i - \bar{Y}) = \Delta X_i \Delta Y_i$ is.



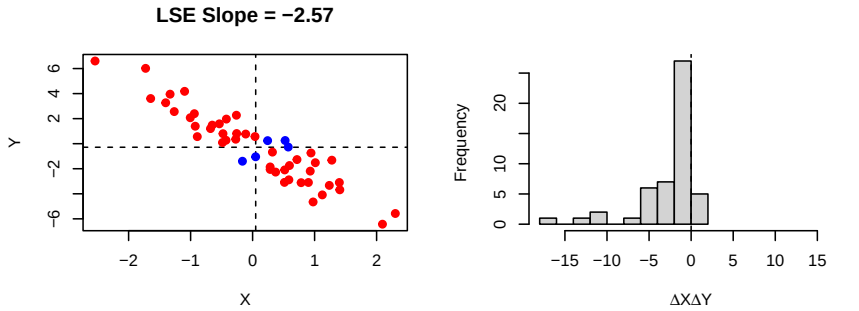
For data that show a positive relationship, many values will lie in quadrants I and III, which both have positive values of the product of ΔX and ΔY. The more data points in these quadrants, the more evidence that accumulates for a positive relationship.

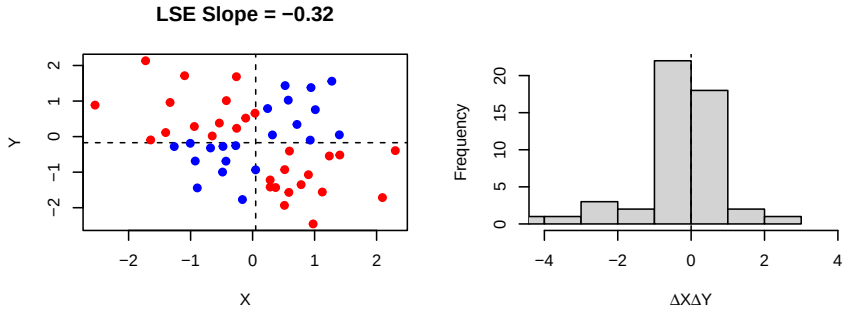
The same argument applies when there are many data points in quadrants II and IV. In that case, most of the data points with have ΔX and ΔY values that are opposite in sign and so will show up as evidence of a negative relationship.

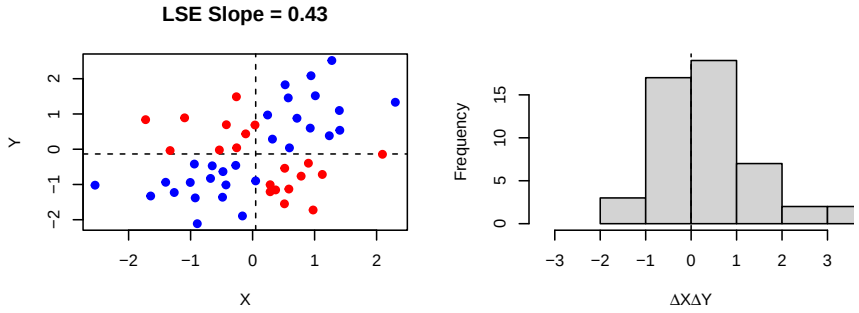
For the *Iris* dataset, it is clear that the relationship is positive. This is because most of the datapoints lie in quadrants I and III. The figure below colors the data points depending on the sign of $\Delta X \Delta Y$. Those with positive sign are in blue and those with negative sign are colored red. You can see that the vast majority are blue, indicating a lot of evidence of a positive relationship. The histogram on the right also shows the distribution of the $\Delta X \Delta Y$ values for all the data points in the data set. The LSE of the slope uses the average of these values.

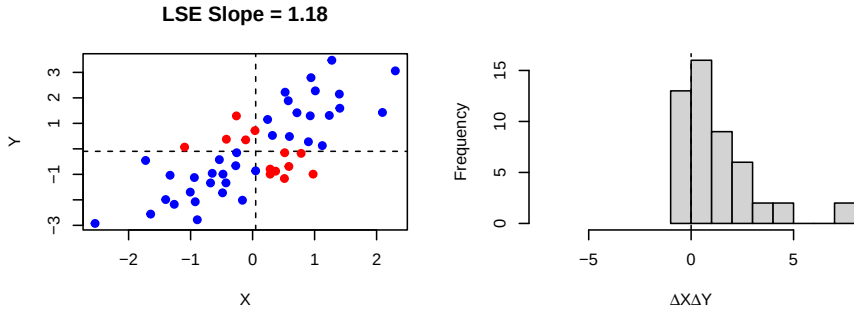


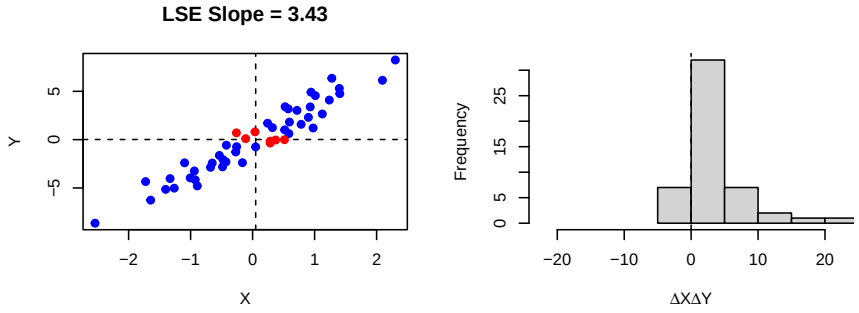
Here is the same kind of figures for other kinds of data with different relationships.

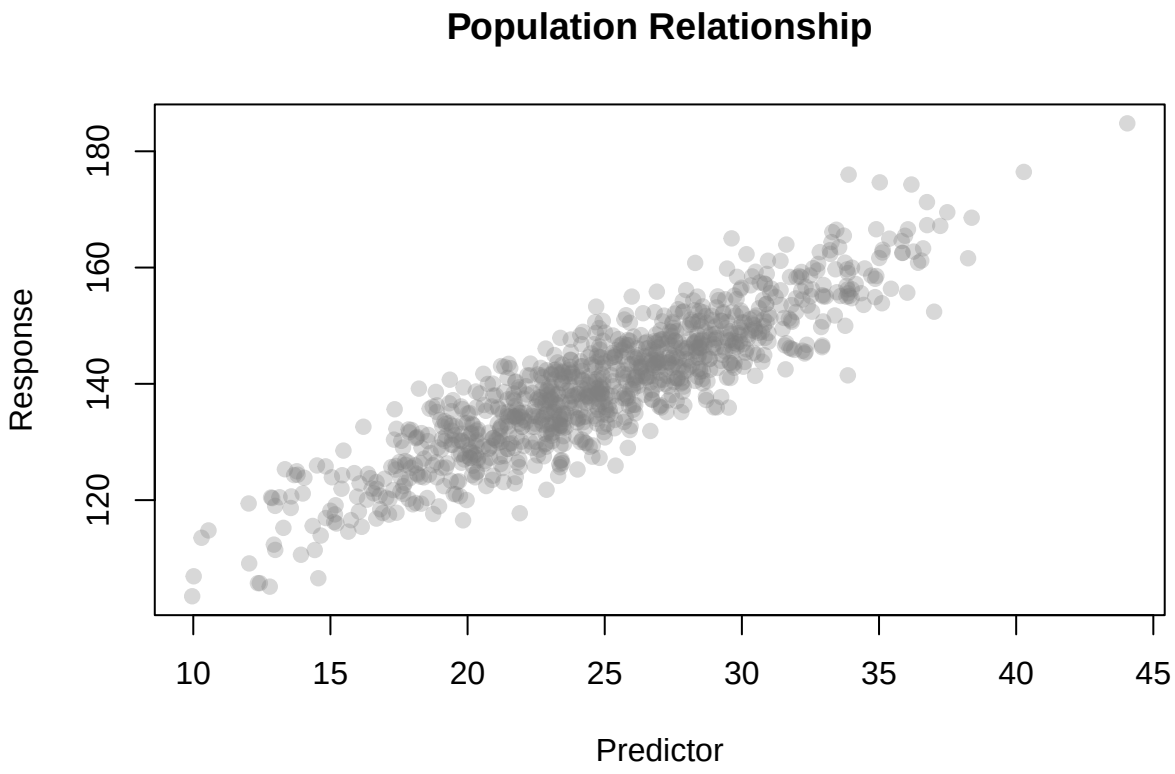

LSE Slope = −2.57
Y
X
Frequency
ΔXΔY


LSE Slope = −0.32
Y
X
Frequency
ΔXΔY


LSE Slope = 0.43
Y
X
Frequency
ΔXΔY


LSE Slope = 1.18
Y
X
Frequency
ΔXΔY


LSE Slope = 3.43
Y
X
Frequency
ΔXΔY

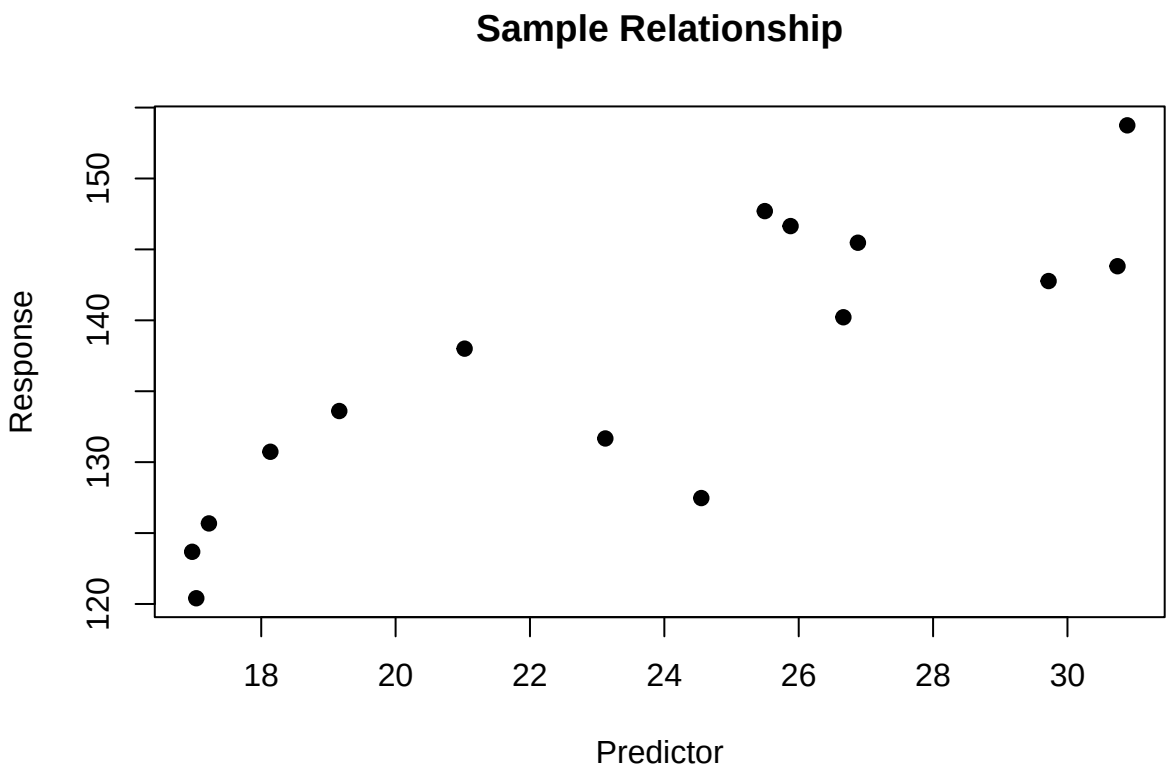


A sample

```
samp.n <- 15
samp.ind <- sample(1:n, samp.n, replace = T)

data <- data.frame(X[samp.ind], Y[samp.ind])
colnames(data) <- c('Predictor', 'Response')

head(data)
```

```
##   Predictor Response
## 1  29.71918 142.7682
## 2  25.49539 147.7000
## 3  25.87871 146.6428
## 4  30.74480 143.8145
## 5  17.03357 120.4046
## 6  23.12171 131.6691
```

```
plot(data$Predictor, data$Response,
     col = 'black', pch = 19,
     xlab= 'Predictor', ylab = 'Response',
     main = 'Sample Relationship')
```

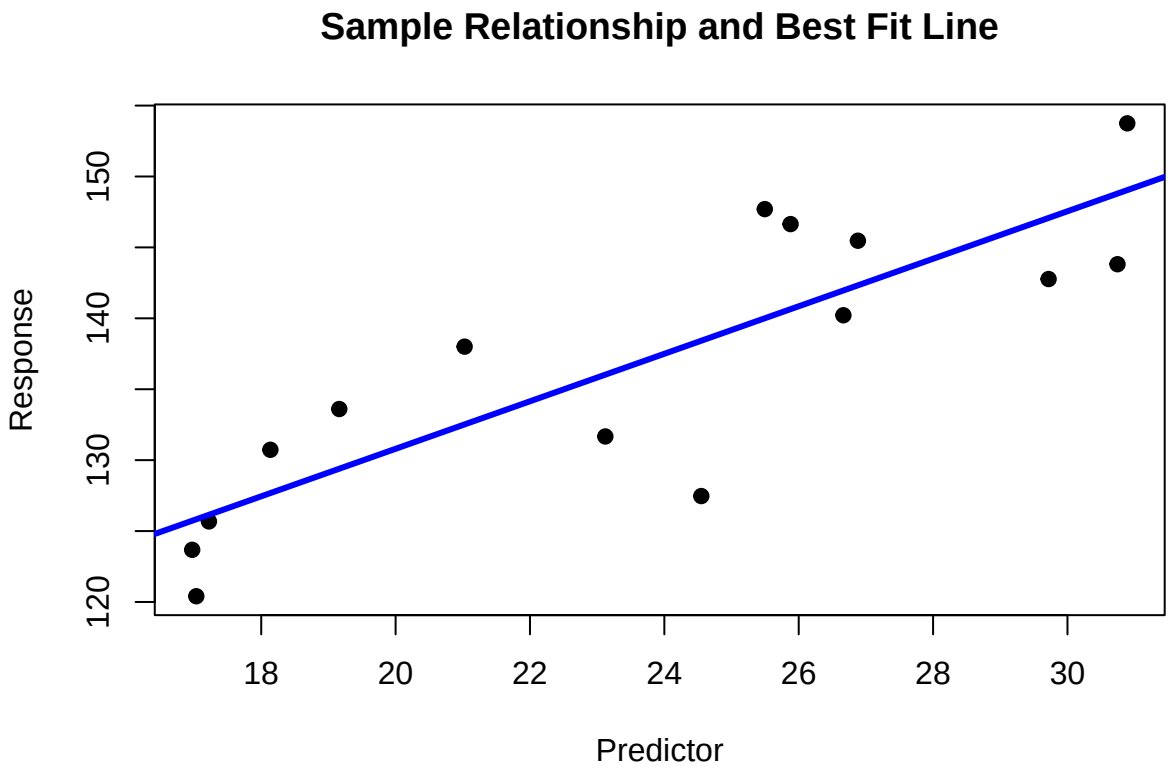


Estimation of Relationship With Linear Regression

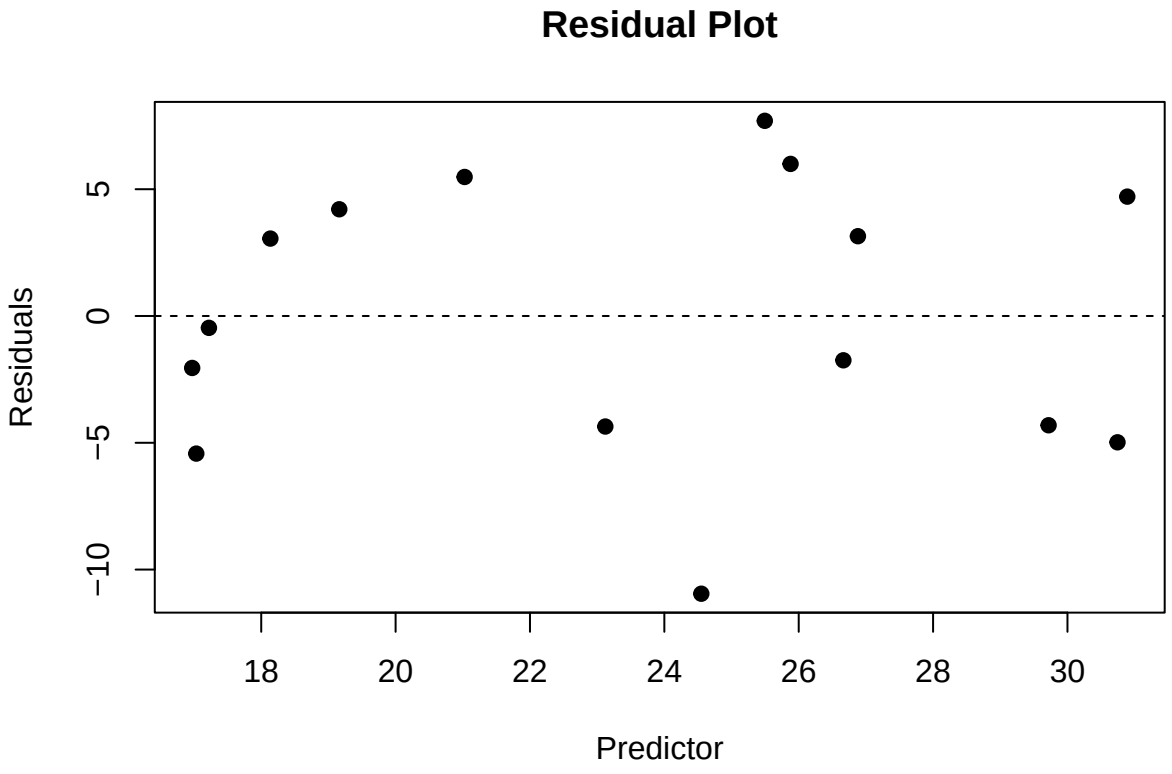
```
lin.mdl <- lm(Response ~ Predictor, data)
summary(lin.mdl)
```

```
##
## Call:
## lm(formula = Response ~ Predictor, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.9520  -4.3340  -0.4677   4.4577   7.6967
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  97.3025     7.1105  13.684 4.26e-09 ***
## Predictor     1.6748     0.2955   5.668 7.69e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.572 on 13 degrees of freedom
## Multiple R-squared:  0.7119, Adjusted R-squared:  0.6898
## F-statistic: 32.13 on 1 and 13 DF,  p-value: 7.691e-05
```

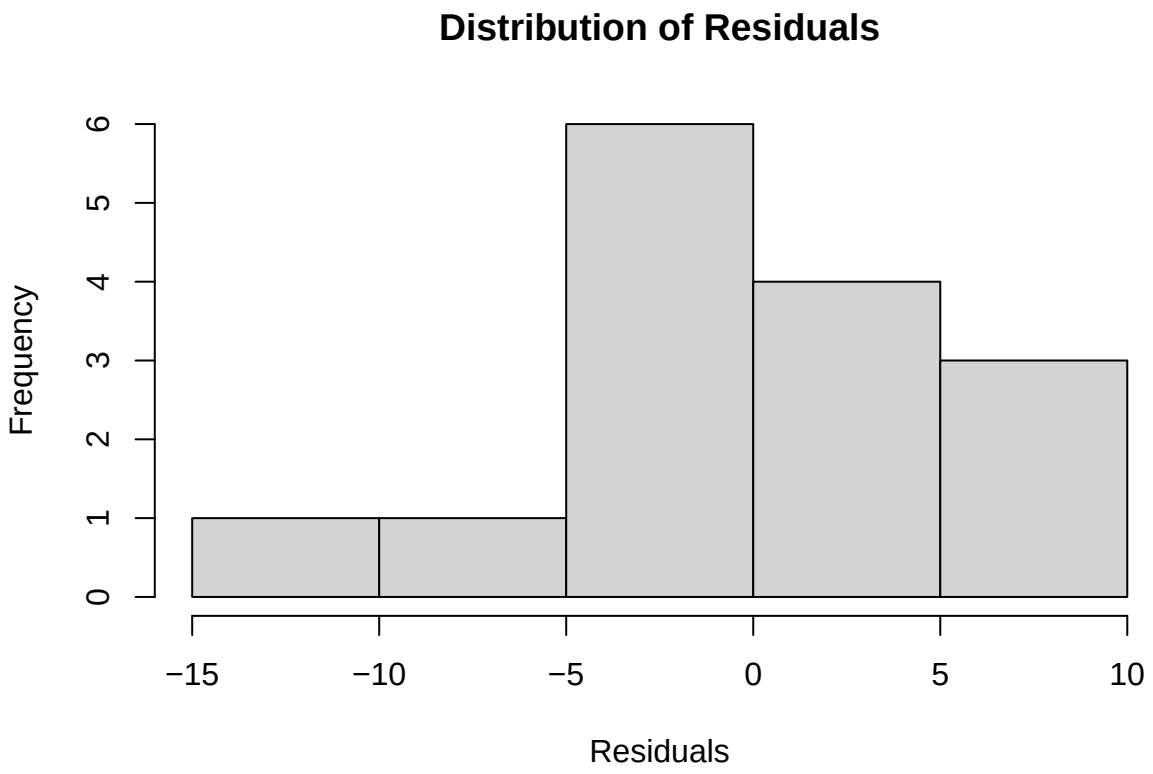
```
plot(data$Predictor, data$Response,
     col = 'black', pch = 19,
     xlab= 'Predictor', ylab = 'Response',
     main = 'Sample Relationship and Best Fit Line')
abline(lin.mdl$coefficients, col = 'blue', lwd = 3)
```




```
plot(data$Predictor, lin.mdl$residuals,
     xlab = 'Predictor', ylab = 'Residuals',
     main = 'Residual Plot', pch = 19)
abline(h = 0, lty = 2)
```



```r
hist(lin.mdl$residuals, xlab = 'Residuals',
     main = 'Distribution of Residuals')
```




```r
(CI95 <- confint(lin.mdl))
```

```
##                2.5 %    97.5 %
## (Intercept) 81.94115 112.66386
## Predictor    1.03650   2.31319
```

```r
est.cov <- vcov(lin.mdl)
```

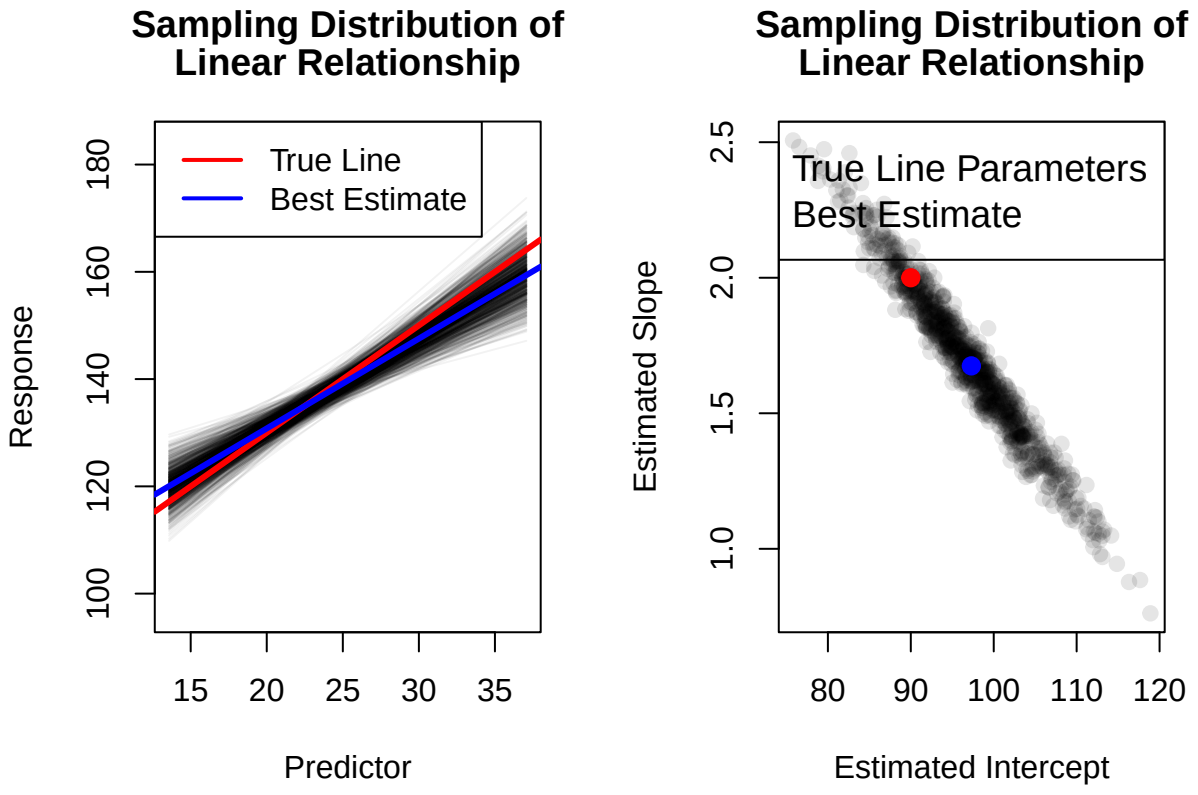
Generalization of Confidence Intervals

Confidence intervals don't apply just to the means, but to any properties that you might be interested in about a population. A natural extension you have likely seen before is the relationship between two measurements about single individuals in a population. For example, you might measure the height of a plant and think that is related to the number of seeds that are produced. We will assume this is the case and create some data on seed set (number of seeds produced by the plants) for each of the plants.

Turns out with the model output, we can find the sampling distribution for this relationship. One way to visualize the sampling distribution is the following plot. Each point shows a different sample of the population and the estimated line associated with it

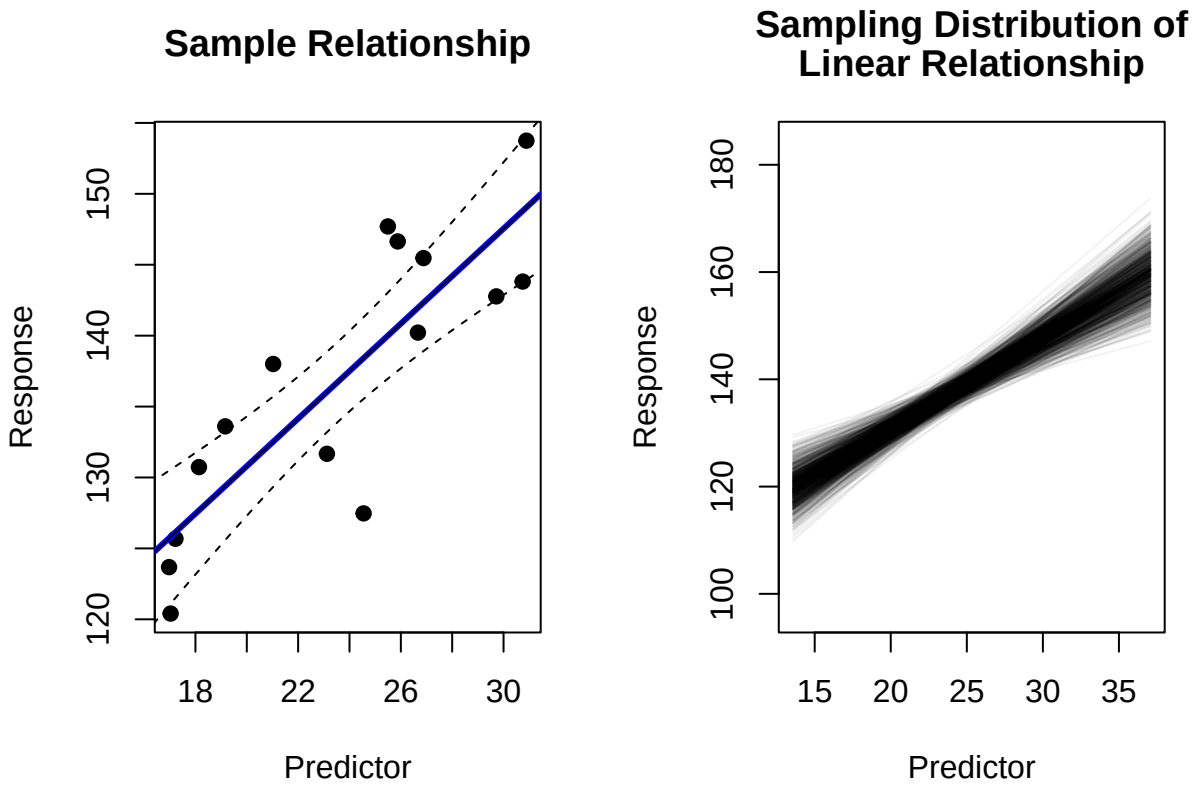
```r
library(MASS)
post.samples <- 1000
rand.estimates <- mvrnorm(post.samples,
                          mu = coefficients(lin.mdl),
                          Sigma = est.cov) # from MASS package
par(mfcol = c(1,2))
x <- seq(from = 0.8*min(data$Predictor),
         to = 1.2*max(data$Predictor),
```

```
        length = 1000)
mean.predict <- coefficients(lin.mdl)[1] + coefficients(lin.mdl)[2]*x
plot(0,0, 'n', xlab = 'Predictor', ylab = 'Response',
     main = c('Sampling Distribution of', 'Linear Relationship'),
     xlim = c(0.8*min(data$Predictor), 1.2*max(data$Predictor)),
     ylim = c(0.8*min(data$Response),
              1.2*max(data$Response)),
     cex = 1.2)
for (i in 1:post.samples){
  pred.mdl = rand.estimates[i,1] + rand.estimates[i,2]*x
  lines(x, pred.mdl, col = rgb(0,0,0, alpha = 0.05))
}
abline(true.int, true.slope, col = 'red', lwd = 3)
abline(lin.mdl$coefficients, col = 'blue', lwd = 3)
legend('topleft', legend = c('True Line', 'Best Estimate'),
       col = c('red', 'blue'), lty = 1, lwd = 2)

plot(rand.estimates[,1], rand.estimates[,2], xlab = 'Estimated Intercept',
     ylab = 'Estimated Slope', pch = 19, col = rgb(0,0,0, alpha = 0.1),
     main = c('Sampling Distribution of', 'Linear Relationship'))
points(true.int, true.slope, col = 'red', pch = 19, cex = 1.2)
points(lin.mdl$coefficients[1],lin.mdl$coefficients[2], col = 'blue', pch = 19, cex = 1.2)
legend('topright', legend = c('True Line Parameters', 'Best Estimate'),
       col = c('red', 'blue'), pch = 19, cex = 1.2)
```



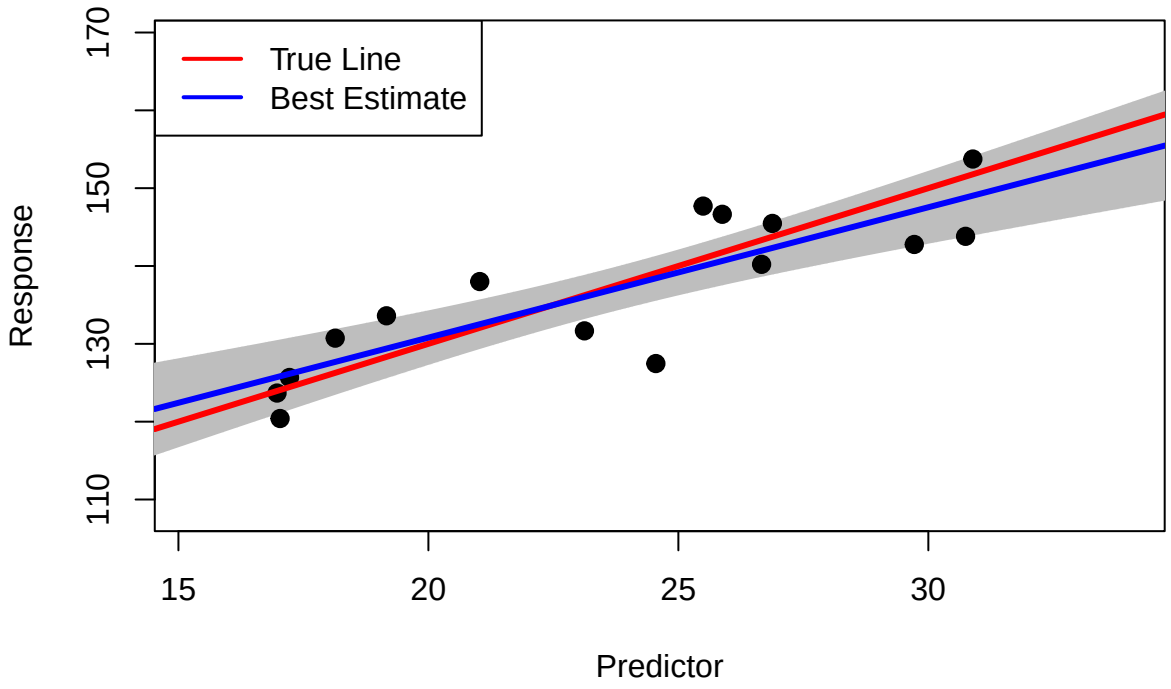

```
# We can find the 95% confidence interval for the regression relationship, yhat.
# It is itself a random variable that is normally distributed conditional on x.
# The mean is
mu_coef <- coefficients(lin.mdl)
```

```r
Xbar_yhat <- mu_coef[1] + mu_coef[2]*x

# And the standard deviation is
SE_yhat <- sqrt(est.cov[1,1] + x^2*est.cov[2,2] + 2*x*est.cov[1,2])

# The 95% confidence interval for each x value is then
yhatCI025 = Xbar_yhat + qt(0.025, n-2)*SE_yhat
yhatCI975 = Xbar_yhat + qt(0.975, n-2)*SE_yhat

par(mfcol = c(1,2))
plot(data$Predictor, data$Response,
     col = 'black', pch = 19,
     xlab= 'Predictor', ylab = 'Response',
     main = 'Sample Relationship')
abline(lin.mdl$coefficients, col = 'blue', lwd = 3)
lines(x, Xbar_yhat)
lines(x, yhatCI025, lty = 2)
lines(x, yhatCI975, lty = 2)

plot(0,0, 'n', xlab = 'Predictor', ylab = 'Response',
     main = c('Sampling Distribution of', 'Linear Relationship'),
     xlim = c(0.8*min(data$Predictor), 1.2*max(data$Predictor)),
     ylim = c(0.8*min(data$Response),
              1.2*max(data$Response)),
     cex = 1.2)
for (i in 1:post.samples){
  pred.mdl = rand.estimates[i,1] + rand.estimates[i,2]*x
  lines(x, pred.mdl, col = rgb(0,0,0, alpha = 0.05))
}
```



This was sort of the round about way to find it. Here's a better way to do it.

```
plot(0,0, typ = 'n', xlab = 'Predictor', ylab = 'Response',
     main = '',
     xlim = c(0.9*min(data$Predictor), 1.1*max(data$Predictor)),
     ylim = c(0.9*min(data$Response),
              1.1*max(data$Response)),
     cex = 1.2)
polygon(c(x,x[length(x):1]), c(yhatCI975, yhatCI025[length(x):1]),
        col = 'gray', border = NA)
points(data$Predictor, data$Response, pch = 19, cex = 1.2)
abline(true.int, true.slope, col = 'red', lwd = 3)
abline(lin.mdl$coefficients, col = 'blue', lwd = 3)
legend('topleft', legend = c('True Line', 'Best Estimate'),
       col = c('red', 'blue'), lty = 1, lwd = 2)
```


True Line
Best Estimate
Response
Predictor
110
130
150
170
15
20
25
30