Lines and functions

Adam Tallman

2022-11-21

Understanding functions

A function is an expresion or rule or law that defines a relationship between one variable and another. In the contexts of statistics we usually think of one of these variables are being the independent variable and the other as being the dependent variable. Usually this translates as follows.\

Dependent variable: the thing you are predicting from one or more independent variables \

Independent variable: the thing you use to predict something about your dependent variable.

Let's we are interesting in seeing whether the slower acquisition of passive sentences in English speaking children can be related to the frequency with which they are exposed to passives. We design an experiment with a treatment effect - being exposed to more passives. The control group are the children who are not exposed. \

In this case the dependent will have to be some metric of how well the children know passives. The independent is our treatment, a trial whereby we teach them passives. \

Doing statistics will involve using a model which predicts how well each child will improve in their passives after the training (Brooks & Tomasello 1999 "Young children learn to produce passives with none verbs", *Developmental psychology*). \

Or say we are interested in knowing whether the size of a speech community influences how many phonemes is in the language of that speech community. In this case, population size will be the independent variable and the phoneme inventory size will be the dependent variable (Moran et al. 2012 "Revising population size vs. phoneme inventory size", *Language*). \

When we think in terms of causation, the whole purpose of running a statistical model is usually to try to argue that our dependent variable *causes* our independent variable. When we think really hard about

In abstract form functions are normally written as follows and we say y is equal to f of x.

$$y \leftarrow f(x)$$

All this means is that you can predict what y is based on x. You should also be able to do the reverse from going backwards. So for instance, we can say celcius is a function of farenheit.

```
fahrenheit_to_celsius <- function(temp_F) {
  temp_C <- (temp_F - 32) * 5 / 9
```

```
  return(temp_C)
}
```

The second line in the code above defines the function - its like writing the following in math.

$$C \leftarrow (F - 32) \times \frac{5}{9}$$

Now put any number in Fahrenheit and it will return a number in Celsius. For example 32 degrees Fahrenheit is 0 degrees celcius.

```
fahrenheit_to_celsius(32)

## [1] 0
```

As I stated above the second line of the R function() code expresses the function from y (celcius) to x (farenheit). A function can also be understood geometrically. Often when I write a function on the board I draw a geometric image that expresses what the relationship is like as well. Its quite easy to express this relationship in R. We just create a set of Farenheit values.

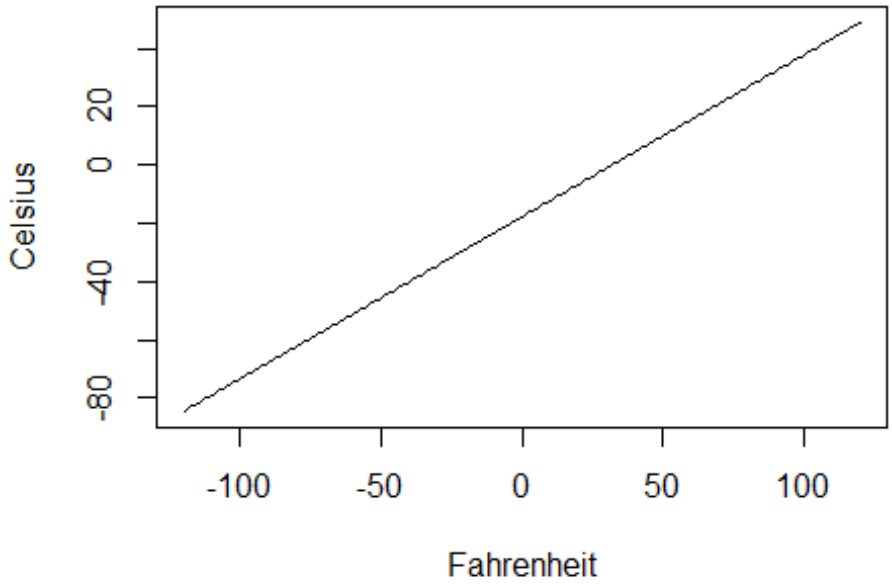
```
F_temps <- seq(-120,120)
```

And how do we create celcius values from these Farenheit values? Well we take that function and apply it over all the values in F_temps.

```
C_temps <- (F_temps - 32) * 5 / 9
```

Then we can plot the relationship between celcius and Farenheit.

```
plot(F_temps, C_temps, type="l", ylab="Celsius", xlab = "Fahrenheit" )
```



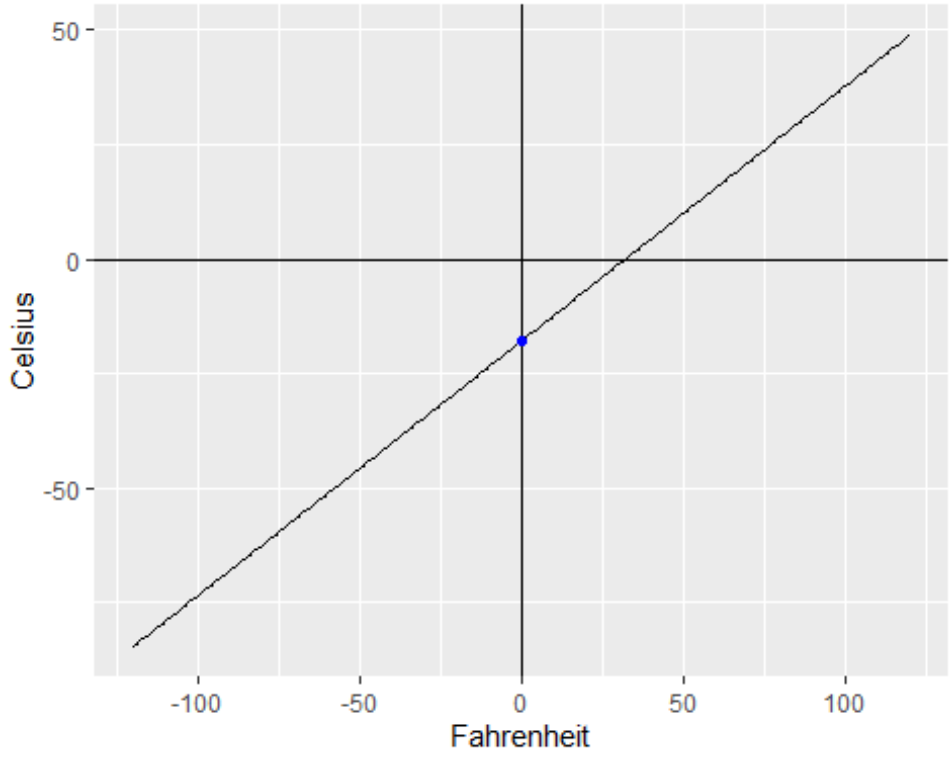
Often we use a graphical display like above to mean the same thing as - usually when the math is too hard or not particularly relevant I will draw the relationship on the board.

$$C \leftarrow (F - 32) \times \frac{5}{9}$$

There are two important concepts for understanding a line. The first is the intercept: this is where the line cross 0 on the y axis. The second is the slope. This is a coefficient that tells how much a change in y gives a change in x. \

We can use a *Cartesian plane* to easily see where the intercept is. A cartesian plane is x=y plot where lines cross 0 on the x and y axis.

```
df <- data.frame(F_temps, C_temps)
ggplot(df, aes(F_temps,C_temps))+geom_line()+geom_hline(yintercept=0)+geom_vl
ine(xintercept=0)+
  xlab("Fahrenheit")+
  ylab("Celsius")+
  annotate("point", x = 0, y =-17.7778 , colour = "blue")
```



The little blue dot there is the intercept. It is where you line crosses the line crosses the y axis at 0. If you want to know the intercept and the coefficient for a line equation you can put the numbers through line().

```
line(C_temps~F_temps)

##
## Call:
## line(C_temps ~ F_temps)
##
## Coefficients:
## [1]  -17.7778    0.5556
```

-17.7778 is the intercept and 0.5556 is the slope.

Note that the data provided above translates to the following equation.

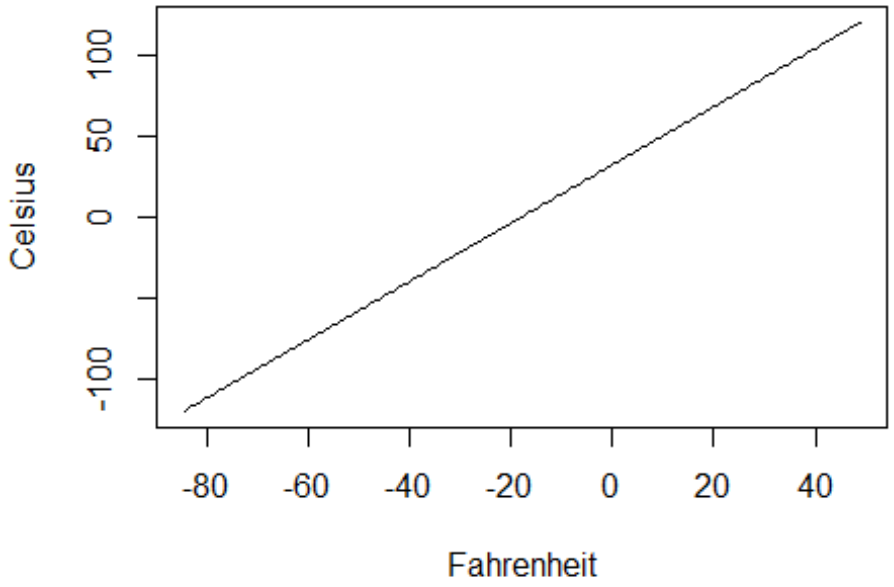
$$ Celcius \space values \leftarrow -17.7778 + 0.556 \times Fahrenheit \space values $$

More abstractly we can write the following in R.

```
#y <- -17.7778 + x*0.5556
```

Now let's generate a set of x values and generate y values from that value based on the formula above, and then plot it again.

```
x <- seq(-120, 120)
y <- -17.7778 + x*0.5556
plot(y, x, type="l", ylab="Celsius", xlab = "Fahrenheit" )
```



It's the same as before. How? Well we created values according to the formula for Fahrenheit and Celsius and then asked R to plot them. It turns out this relationship is a linear. Then we asked R to calculate a line with the typical formula for a line which is

$$y \leftarrow a + b \times x$$

where a is the intercept and b is the slope. We reproduced the y values again from this formula. Actually really all this means is that:

$$-17.7778 + 0.556 \times F = (F - 32) \times \frac{5}{9}$$

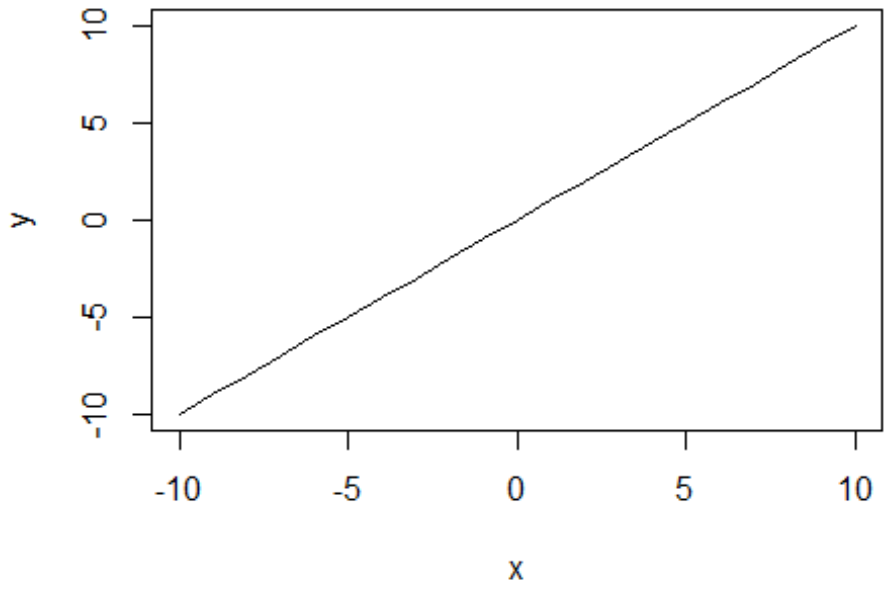
But note that the form on the left side of the equation is the form given for a line. When R gives you an intercept and a coefficient it gives you the form of the equation on the left (note, if you actually try to calculate numbers you'll find that the left and right sides of the equation aren;t exactly equal but this is because of floating point errors).

Let's break this down again. The simplest relationship between x and y is equality.

$$y \rightarrow x$$

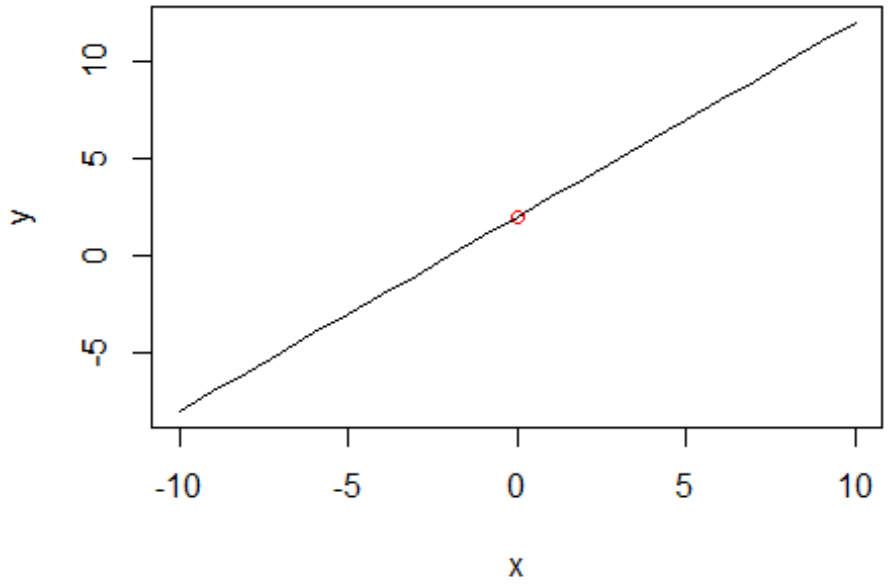
We can plot this as follows:

```
x <- seq(-10, 10)
y <- x
plot(x,y, type = 'l')
```



You can see that for this line the intercept is 0. How do we change the intercept, well if the intercept is 2 we add 2 to the equation.

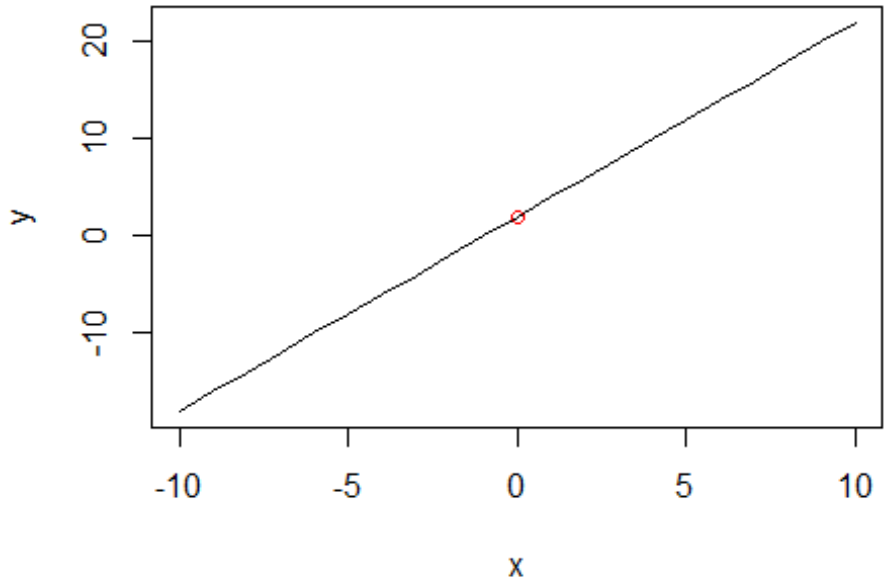
```
x <- seq(-10, 10)
a <- 2
y <- a + x
plot(x,y, type = 'l')
points(0, 2, col = "red")
```



You can see the line crosses on the x axis at 2 y (at the red dot). \

The slope of the line above is 1. If we want to change the slope so that its 2 - then y will be twice of x for every x, plus 2 which is the intercept.

```
x <- seq(-10, 10)
a <- 2
b <- 2
y <- a + b*x
plot(x,y, type = 'l')
points(0, 2, col = "red")
```



Please watch the following video on youtube to get a basic brush up of linear equations.

https://www.youtube.com/watch?v=Ft2_QtXAnh8&t=1149s

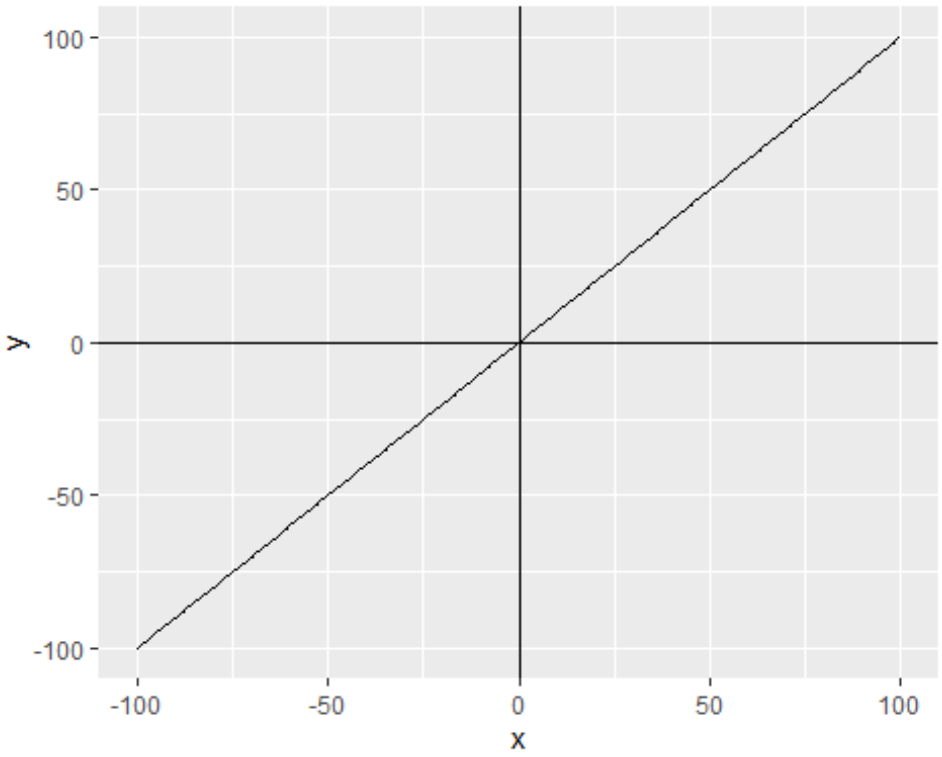
Understanding functions for statistics

The difference between a linear equation/model like the one we gave in the previous exercise and a linear model in statistics (or "regression") is that a linear model in statistics has some "stochastic component". We can use x for the predictor variable, a for the intercept, beta for the coefficient and epsilon for the error term. The error term varies "randomly" around some mean values. Your statistical model needs to calculate how large that number is in order to know how much the linear model is accounting for the relationship between x and y.

$$y \leftarrow a + \beta * x + \epsilon$$

Let us visualize this abstractly (just with numbers) and then we can get into a worked example. First we have the linear model. The model below is not a statistical model.

```
set.seed(222)
x1 <- seq(-100, 100)
b <- 1
a <- 0
y <- a + b*x1
df <- data.frame(y, x1)
ggplot(df, aes(y,x1))+geom_line()+geom_hline(yintercept=0)+geom_vline(xinterc
ept=0)+
  xlab("x")+
  ylab("y")
```

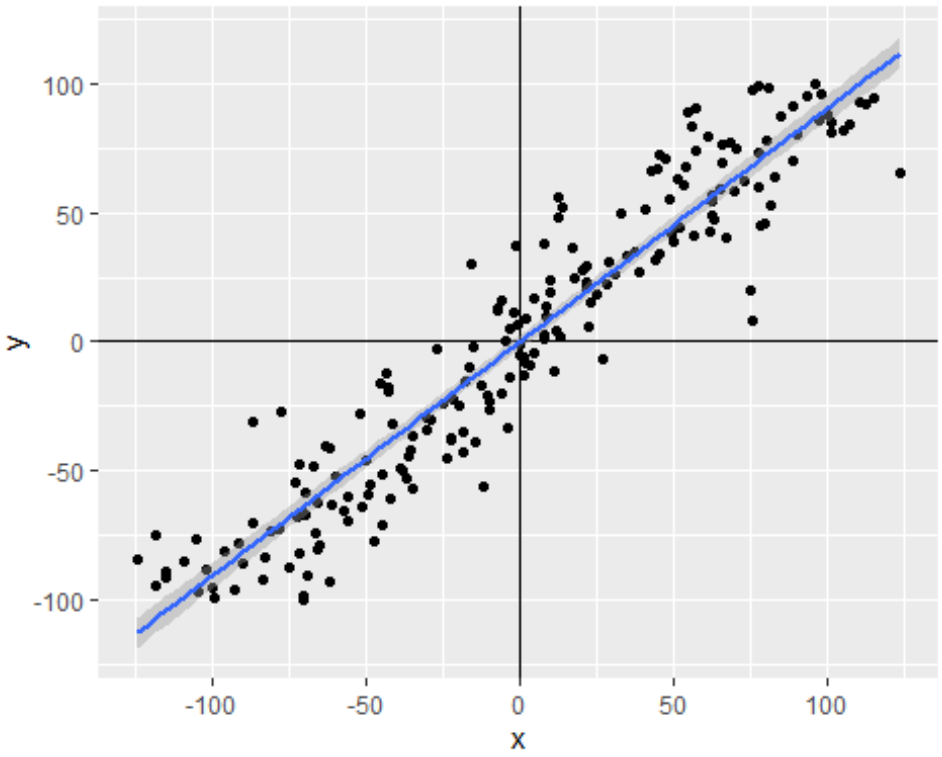


This model just says that y is the same as x. But it has parameters that can be changed in case this isn't true. To make it a statistical model. We add an error term. When we add the error term we are simulating a statistical model with stochasticity.

```
x1 <- seq(-100, 100)
b <- 1 #The coefficient - here we are saying there is no difference between x and y by making it 1.
a <- 0 #The intercept - x is not increased by some fixed value in relation to y.
e <- rnorm(m=0, sd=20, n=201) #We have to make n 201 because that's the numbe r of data points. In the normal case error terms have a mean of 0.
y <- a + b*x1 + e # a is the intercept, b is the coefficient (how much change there is), x1 is the predictor variable, y is the predicted variable, e is th e error term. e is the reason we do statistics at all.
df <- data.frame(y, x1)

ggplot(df, aes(y,x1))+geom_hline(yintercept=0)+geom_vline(xintercept=0)+
  geom_point()+
  xlab("x")+
  ylab("y")+
  geom_smooth(method='lm')

## `geom_smooth()` using formula 'y ~ x'
```



The plot above actually builds a model of y according to x. We can specify this model without plotting it as follows. The function summary() gives you a summary of your linear model.

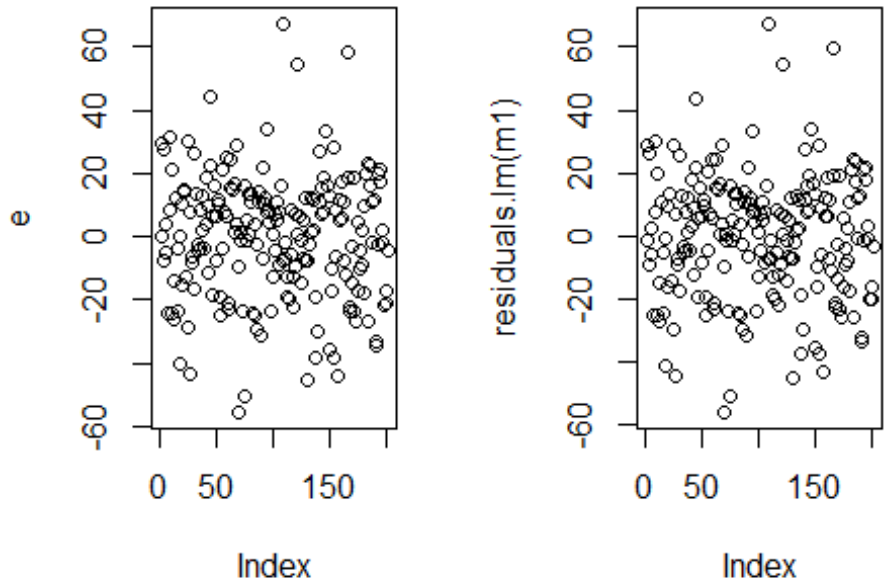
```
m1 <- lm(y~x1) #Build a model call it m1, where y is the predicted variable a
nd x is the predictor variable
summary(m1)

##
## Call:
## lm(formula = y ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.935 -12.787   1.306  12.306  67.361
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.08498    1.39557   0.061    0.952
## x1           0.98837    0.02405  41.093   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.79 on 199 degrees of freedom
## Multiple R-squared:  0.8946, Adjusted R-squared:  0.894
## F-statistic:  1689 on 1 and 199 DF,  p-value: < 2.2e-16
```

Okay the thing you should pay attention to is the coefficient (Estimate). This is 0.99530, which is close to 1. This makes sense because this is what we told the model to do, we made the coefficient for the data 1. The statistical model added 1 for an intercept. These discrepencies are produced through the random error that we added.

What about our error term that we added. Okay, there's two concepts that you will hear about in statistics that are relevant here. The *error* and the *residual*. The residuals are the distances that each of your points are from your model. The error is the deviation from the true value or the population value. The residual is the deviation from what the model says. You can get the residuals of a model by using the function residuals.lm().

```
par(mfrow = c(1, 2))
plot(e)
plot(residuals.lm(m1))
```



We can see that these values are pretty close. They are not exactly the same thing because the model will model a little of randomness.

```
sd(e)

## [1] 19.74761

sd(residuals.lm(m1))

## [1] 19.73603
```

The whole point of a regression model is to fit a line that makes the overall (mean) distance from the line as small as possible. This is going to be reflected in the mean difference between our error and our residuals.

```
mean(e)

## [1] 0.08497738

mean(residuals.lm(m1))

## [1] 1.708282e-16
```

If you square these values the means become much closer.

```
mean(e^2)

## [1] 388.0354

mean(residuals.lm(m1)^2)

## [1] 387.573
```

So there is going to be some discrepancy between the *real* value and the value estimated by the model, but the model gets pretty close.

A typical thing you'll want to look at is a residual plot for your regression model. You want your residuals to have a consistent relationship to your model or be normally distributed around it - otherwise your model might be "misspecified", e.g. the relationship isn't linear, but you assumed that the relationship was a straight line.

A worked example (son height vs father height)

A very obvious type of stochastic relationship is the relationship between the sons' heights and father's heights. Obviously we should expect these to be correlated, but there should be lots of variation because there's all sorts of other things that affect your height, like the height of your mother, environmental factors and a number of other uncontrolled factors.

There's a real dataset where you can find some measurements for son height versus father height. It can be found in the following packages, which you should install before loading.

```
library(Sleuth3) # Contains data for problemset

## Warning: package 'Sleuth3' was built under R version 4.2.2

library(UsingR) # Contains data for problemset

## Warning: package 'UsingR' was built under R version 4.2.2

## Warning: package 'MASS' was built under R version 4.2.2

## Warning: package 'HistData' was built under R version 4.2.2

## Warning: package 'Hmisc' was built under R version 4.2.2

library(MASS) # Modern applied statistics functions
```

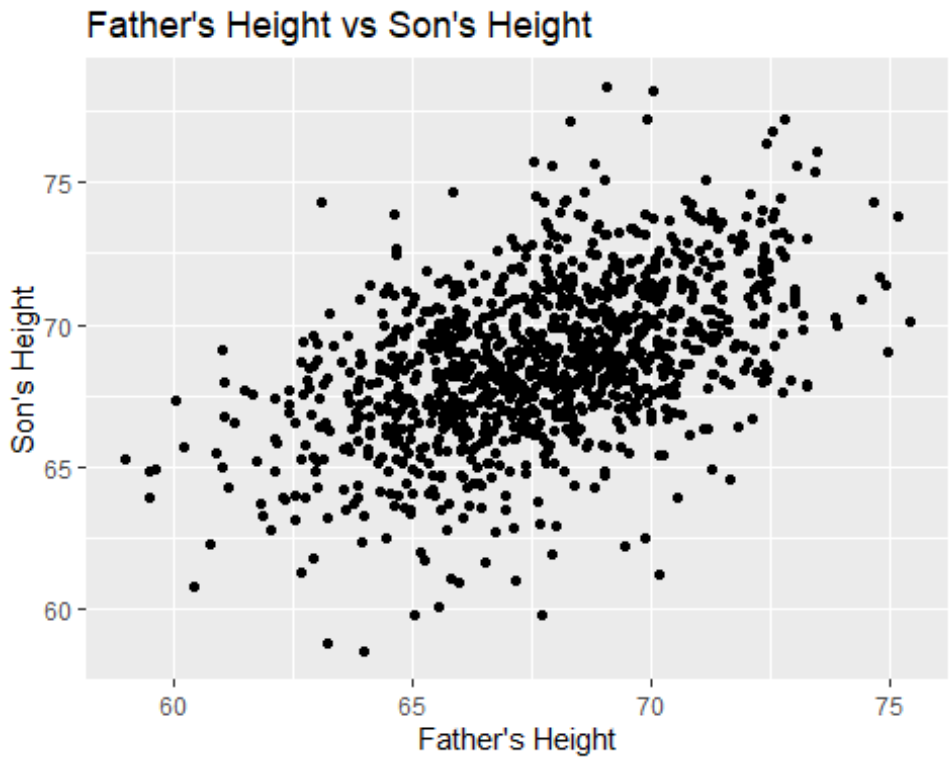
Load the heigh data with the following formula.

```
heightData <- tbl_df(get("father.son"))
```

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
generated.
```

In these data fheight is father height and sheight if the height of their son.

```
ggplot(heightData, aes(x = fheight, y = sheight)) +
  geom_point() +
  #stat_smooth(method = "lm", col = "red")+
  xlab("Father's Height")+
  ylab("Son's Height")+
  labs(title="Father's Height vs Son's Height")
```



You can create a linear model as follows.

```
model.height <- lm(sheight~fheight, data = heightData)
summary(model.height)

##
## Call:
## lm(formula = sheight ~ fheight, data = heightData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8772 -1.5144 -0.0079  1.6285  8.9685
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.88660    1.83235   18.49   <2e-16 ***
## fheight      0.51409    0.02705   19.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.437 on 1076 degrees of freedom
## Multiple R-squared:  0.2513, Adjusted R-squared:  0.2506
## F-statistic: 361.2 on 1 and 1076 DF,  p-value: < 2.2e-16
```

The intercept is 33.88660 and the coefficient is 0.51409. This means that in order to get the father's height from the son's height at 33.9 cm and multiple the father's height by 0.5.

The R-squared is a measurement of how much the father's height is accounting for the son's height. Its saying that about 25% of the variation is explained by the father's height.

The traditional way of reporting this would be as follows:

"There is a statistically significant relationship between the father's height and the son's height (B = 0.51409, R^2 = 0.25, p < 0.0001)"

A less traditional way, but one that actually reports what the model says would be the following.

"A son's height is his father's height plus 33.88600 cm in addition to this for every centimeter increase in the father's height the son's height increases 0.51409 cm (B = 0.51409, R^2 = 0.25, p < 0.0001)"

Or you could say

You could also report the confidence intervals for the coefficient, remember you get this value with the function confint().

```
confint(model.height)
```

```
##                 2.5 %     97.5 %
## (Intercept) 30.2912126 37.4819961
## fheight      0.4610188  0.5671673
```

You could report this by saying:

"The sample mean of the intercept is 33.88660 cm with a 95% confidence of [30.29 cm, 37.48]"

"The height of a father's son is the intercept plus his height times 0.5671673 with a 95% confidence interval of [0.4610188 cm, 0.5671673 cm]"

This is a less conventional way of reporting the model, but more informative than just the p values.

References

Brooks, P. a. (1999). Young children learn to produce passives with nonce verbs. *Developmental Psychology, 35*(1), 29-44.

Moran, S. a. (2012). Revisiting population size vs. phoneme inventory size. *Language, 88*(4), 877-893.