Introduction to Linear Regression

Michael F. Meyer

12 May 2021

To the Reader:

This document is meant to serve as an introduction to linear regression. It is not meant to detail the statistical background behind linear regression, but rather offer a tutorial on how to make and visualize a linear regression in R.

Get the data

The code below is used to load the data necessary for this tutorial. We won't talk about what this code means here, but we will cover those details in the workshop.

```
install.packages("palmerpenguins")
library(palmerpenguins)

penguin_data <- palmerpenguins::penguins
```

What's a linear regression?

Linear regression is among one of the most common statistical tools in the social and natural sciences. In essence, you can think about it as drawing a line through points, so as to define a relationship between at least two variables. To wrap our heads around this, you can think about the generalized equation $y = mx + b$, where y is your response, m is your slope, x is your predictor, and b is the intercept. We use this same structure in a linear regression, but sometimes the notation can be a little different. In a linear regression, we say $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$. The concept is the same between the two forms, but in the second form y_i corresponds to a given observation of our response variable, β_0 is our intercept, β_1 is our slope, x_i is a given observation of our predictor variable, and ϵ_i is an error term.

We'll talk about this model framework more in depth during the statistical analysis portion of the workshop, but it's good to see these formulations early and often. A good friend of mine from graduate school always say "It's not about intelligence. It's just the number of times you've seen it." So, if you don't fully understand everything now, that's 100% okay. You will definitely see it many more times.

How to format data for a linear regression

When building a linear regression (i.e., feeding in data and letting the computer calculate the slope, intercept, etc.), we need to have the data structured in a certain format. In R, this format is often referred to as tabular or "tidy" data, where each row is an observation and each column is a variable. As it relates to our formula above, you can think of y_i and x_i as examples of this concept, where x and y are variables that we may be measuring and the `i` subscript refers to a particular measurement of a given variable.

So, let's look at an example. Here's a dataset from the U.S. Long Term Ecological Research program at the Palmer Station in Antarctica.

```
head(penguin_data)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Adelie  Torge~           39.1          18.7              181        3750 male
## 2 Adelie  Torge~           39.5          17.4              186        3800 fema~
## 3 Adelie  Torge~           40.3          18                195        3250 fema~
## 4 Adelie  Torge~           NA            NA                 NA          NA <NA>
## 5 Adelie  Torge~           36.7          19.3              193        3450 fema~
## 6 Adelie  Torge~           39.3          20.6              190        3650 male
## # ... with 1 more variable: year <int>
```

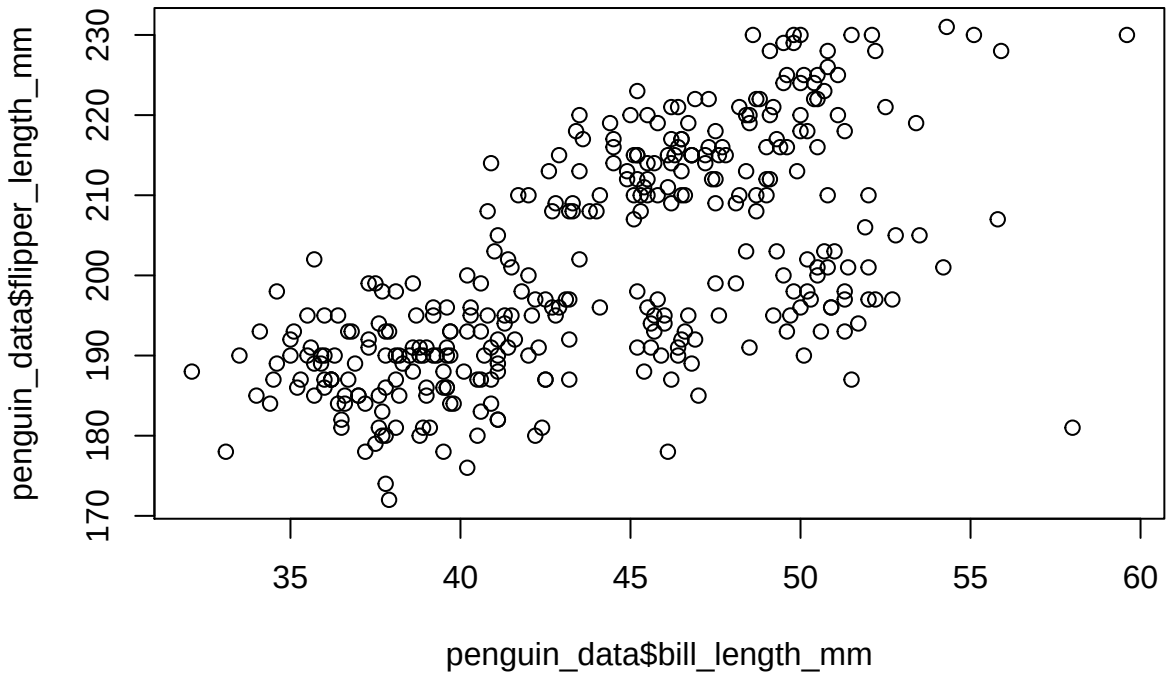
We see that there are several variables contained within this dataset (`species`, `island`, `bill_length_mm`, `bill_depth_mm`, `flipper_length_mm`, `body_mass_g`, `sex`, `year`). Remember, you can think of these as representing x- and y-axes of a plot, and each row can correspond to the `i` subscript. As currently formatted, these data are ready for analyses.

Making a quick plot

Let's say we are interested in how bill length may be related to flipper length.

It's always a good idea to make a quick plot of your data before we model them. You will learn fancy ways of making plots during the workshop, but here is a quick way to visualize the data first.

```
plot(x = penguin_data$bill_length_mm,    # Plot bill length on x-axis
     y = penguin_data$flipper_length_mm) # Plot flipper length on y-axis
```



In general, it looks like penguins with longer bills tend to have longer flippers. Biology tells us that this pattern makes sense because bodyplans tend to scale and be proportionate to one another (especially for vertebrates).

Building a linear regression in R

Building a linear regression in R uses the general format `model_object <- lm(formula = y ~ x, data = data)`. Let's look at this code in pieces. `lm(...)` is the piece that actually makes the formula that we talked about above. The `formula =` parameter is where we define our predictor and response variable. For me, I like to read the `~` as "is a function of". So, in the toy example, we can read the formula as "y as a function of x" or in our penguin example "flipper length as a function of bill length". `data =` specifies the dataframe that R should use to build the model. Lastly, `model_object <-` specifies that R should build the model and store the results in a new variable, which we are calling `model_object` (we can call it anything, though, even something wild like `dogs_riding_horses <- lm(y ~ x)`).

So, let's build the model with the penguin data.

```
penguin_model <- lm(formula = flipper_length_mm ~ bill_length_mm,
                    data = penguin_data)

# We can get the information from the model, using summary()

summary(penguin_model)
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ bill_length_mm, data = penguin_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.708  -7.896   0.664   8.650  21.179
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    126.6844     4.6651   27.16   <2e-16 ***
## bill_length_mm   1.6901     0.1054   16.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.63 on 340 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.4306, Adjusted R-squared:  0.4289
## F-statistic: 257.1 on 1 and 340 DF,  p-value: < 2.2e-16
```

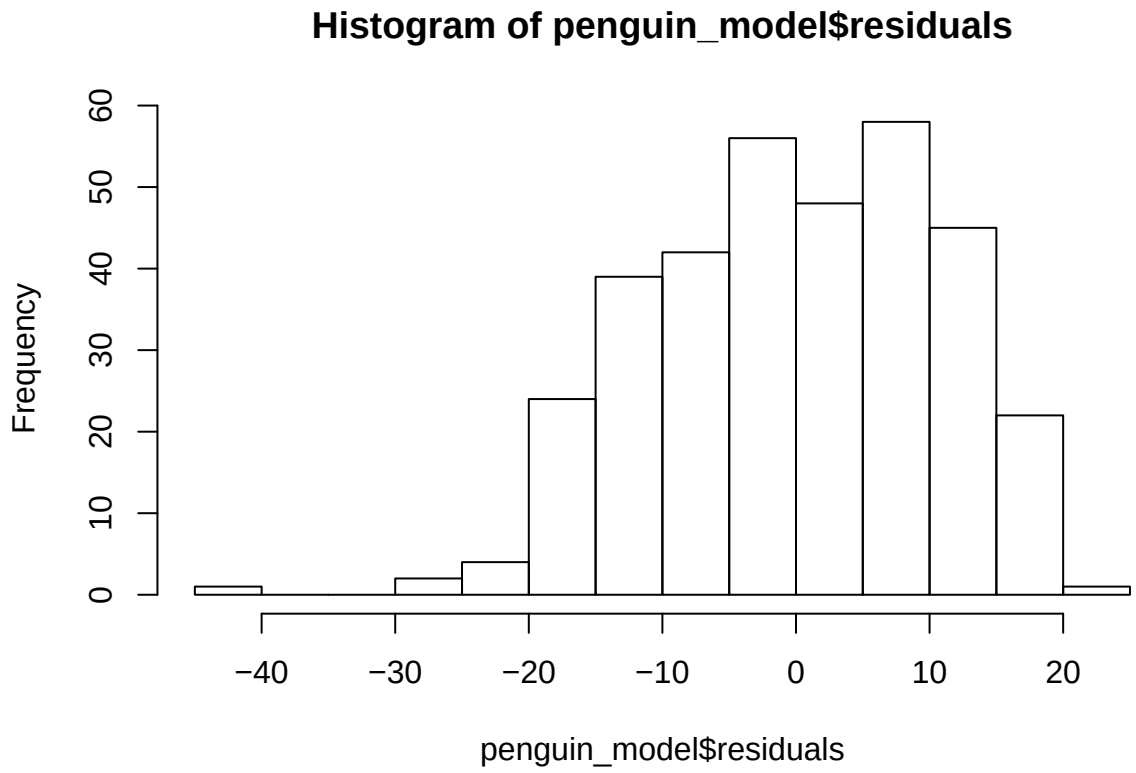
Understanding the output

Now that we've built our model, let's understand the outputs. There's a lot here, but let's highlight the big stuff:

- **The `Estimate` column**: There's an `Intercept` and `bill_length_mm` estimate value. Each of these corresponds to our y-intercept (β_0) and slope (β_1).
- **The R-squared values**: These correspond to how well the model fits our data. A value closer to 1 is really good. A value closer to 0 is not great. Environmental data tend to be very messy, so a R-squared value of ~0.43 is REALLY good.

- **The p-value**: Let's be careful with this one (and we'll talk about why in the workshop), but we can broadly think of a p-value as the probability of observing a given relationship. Here, we see that the p-value is quite low, so we can be pretty confident that this is a non-random process. Biology also says this process should be non-random, too.
- **The Residuals**: This is one of the most important parts of the anlysis. They correspond to the error term (ϵ_i) from our equation above, and they help us know how much our regression deviates from what we actually observed. If we want to visualize our residuals, I prefer looking at a histogram format. You can make a histogram of your residuals, using the command `hist(...)`.

```
hist(penguin_model$residuals)
```

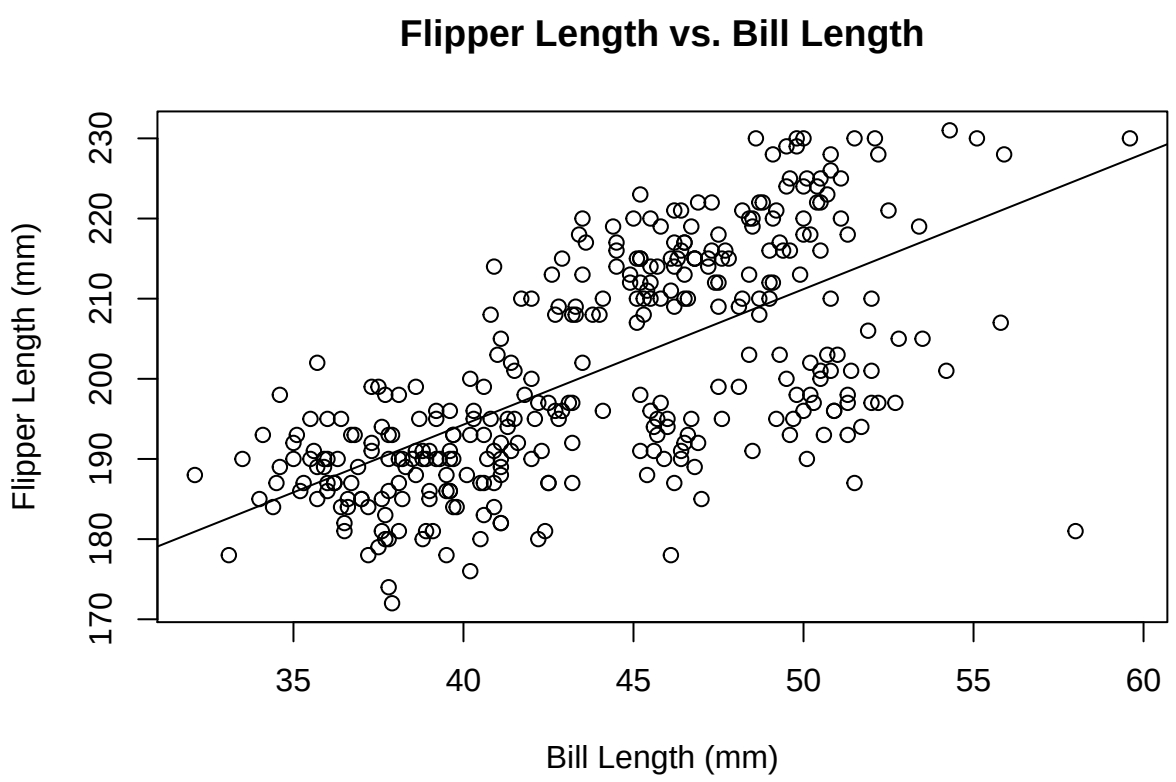


So, it looks like our regression has error terms concentrated around 0 (which is good). Some of the errors extend really far away (e.g., < -20), but those occurrences are pretty rare.

Visualizing the model and data

Lastly, let's plot the data and overlay our regression line. Again, there are much fancier ways of doing this, which you will learn in the workshop, but let's stick to the basics here.

```
plot(x = penguin_data$bill_length_mm,          # This is your x-axis
     y = penguin_data$flipper_length_mm,       # This is your y-axis
     main = "Flipper Length vs. Bill Length", # Give the plot a title
     ylab = "Flipper Length (mm)",             # Set the y-axis label
     xlab = "Bill Length (mm)")                # Set the x-axis label
abline(penguin_model)                          # Add the regression line
```



Conclusion

So, just to recap - we were able to build a regression model in R to explore the relationship between two variables. We'll talk about these topics in more detail during the workshop, but hopefully you gained some familiarity with the main commands to make these regressions.

Questions, comments, or concerns?

If you have questions, comments, or concerns about this tutorial, you can reach out to Michael Meyer over email (michael.f.meyer@wsu.edu).