On the equivalence of t-tests, anovas, and linear models

Brad Duthie

03/06/2020

These are some notes for practice and discussion in Stirling Coding Club, which follow up on a recent meeting in the Stats Discussion Group at the University of Stirling on a blog post by Jonas Kristoffer Lindeløv. What follows is further demonstration and discussion on linear models in statistics. The goal here is to create a simulated data set, then use it show a bit more clearly the logic underlying linear models in statistics.

Contents

Making up data for heights of two plant species

Let's first make up some data and put it into a data frame. To make everything a bit more concrete, let's just imagine that we're sampling the heights of individual plants from two different species. Hence, we'll have one categorical independent variable, and one continuous dependent variable (plant height). I am just going to make up some data to work with below. The data frame below includes plant height (`height`; since this is a made up example, the units are not important, but let's make them mm) and species ID (`species_ID`). The first 10 plants (each plant is a unique row) are shown below.

height	species_ID
186.72	species_2
237.09	species_2
83.51	species_2
121.60	species_1
174.04	species_1
183.89	species_1
195.81	species_2

height	species_ID
153.98	species_1
94.22	species_1
162.58	species_2

Using the linear modelling approach described by Lindeløv, the above data qualify as a simple regression with a discrete x (`species_ID`). Assuming that both species have equal variances in height, we can use a two-sample t-test in R to test the null hypothesis that the mean height of `species_1` is equal to the mean height of `species_2`. To use `t.test`, we can first create two separate vectors of heights, the first one called `species_1`.

```
species_1 <- plant_data$height[plant_data$species_ID == "species_1"];
```

Below shows `species_1`, which includes the heights of all 55 plants whose `species_ID == "species_1"`.

```
##  [1] 121.60 174.04 183.89 153.98  94.22 195.44 167.52 263.51 230.72 112.22
## [11] 104.13 179.00 227.79  83.35 194.07 166.26 120.47  38.30 197.03 164.20
## [21] 177.72 158.31 159.38 124.82  95.66 177.33 163.96 180.11  49.84 277.94
## [31] 219.87 132.00 174.78 166.47  88.50 190.94 150.66  88.56  76.49  96.10
## [41] 143.61  96.86 194.21 218.08 187.58  66.90 195.18  91.13 136.00 129.61
## [51] 207.73 145.95 140.78   7.12 193.08
```

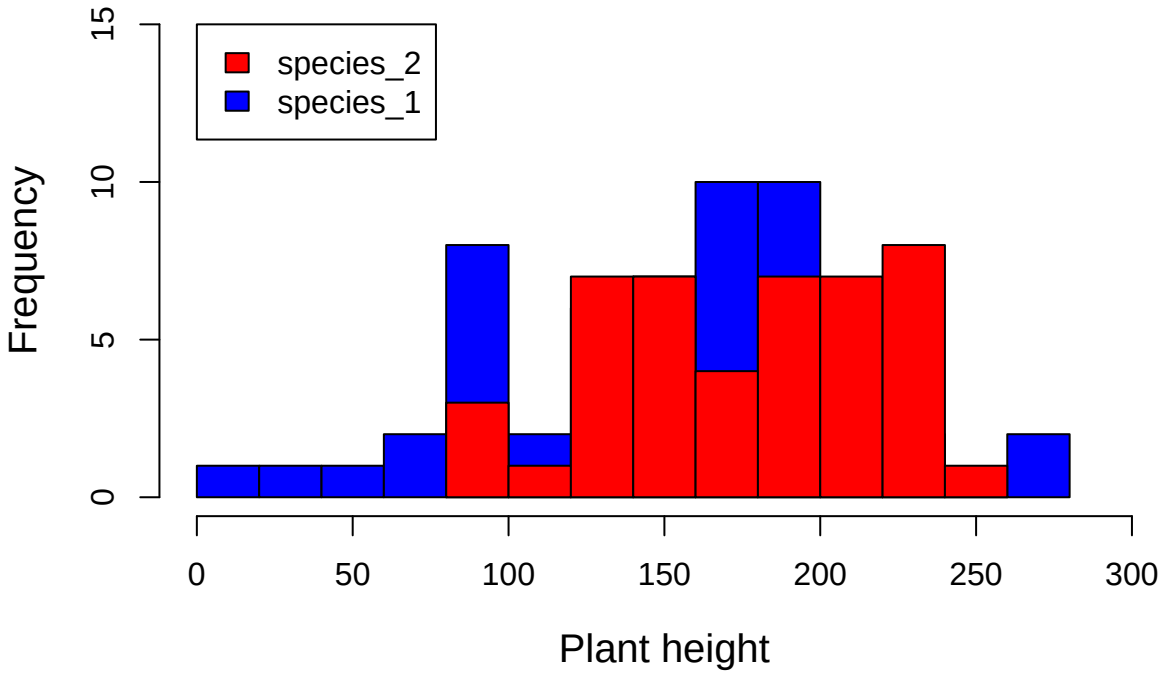
We can make a separate vector for the remaining heights for the plants of species 2 in the same way.

```
species_2 <- plant_data$height[plant_data$species_ID == "species_2"];
```

These 45 plant heights are shown below.

```
##  [1] 186.72 237.09  83.51 195.81 162.58 207.16  83.49 207.36 164.05 148.37
## [11] 131.35 138.02 195.10 171.47  97.87 129.63 227.56 180.62 139.02 221.68
## [21] 139.00 238.26 183.57 123.64 252.44 200.19 171.38 150.35 105.42 229.29
## [31] 226.79 223.13 152.11 147.50 140.20 181.61 128.68 153.50 204.72 200.99
## [41] 230.54 207.57 158.03 209.35 198.66
```

It might help to plot a histogram of the two plant species heights side by side.



Visualising the histogram above, we already have a sense of whether or not knowing species ID is useful for predicting plant height.

Equivalence of `t.test` versus a linear model `lm` in R

Using our two vectors `species_1` and `species_2`, we can run a t-test as noted by Lindeløv.

```
t.test(species_1, species_2, var.equal = TRUE);
```

```
##
##  Two Sample t-test
##
## data:  species_1 and species_2
## t = -2.3997, df = 98, p-value = 0.0183
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -44.453290  -4.210064
## sample estimates:
## mean of x mean of y
##  150.4545  174.7862
```

Reading the output above, we can get the t-statistic `t = -2.3996794`. Given the null hypothesis that the mean height of `species_1` equals the mean height of `species_2`, the probability of getting such an exterme difference between the two observed means is `p-value <` 0.0182991 (i.e., unlikely).

But this is not the only way that we can run a t-test. As Lindeløv points out, the linear model structure

works just fine as well.

```
lmod1 <- lm(height ~ 1 + species_ID, data = plant_data);
summary(lmod1);
```

```
##
## Call:
## lm(formula = height ~ 1 + species_ID, data = plant_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -143.34  -35.77    7.34   33.72  127.48
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           150.455      6.802   22.12   <2e-16 ***
## species_IDspecies_2    24.332     10.140    2.40   0.0183 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.44 on 98 degrees of freedom
## Multiple R-squared:  0.0555, Adjusted R-squared:  0.04586
## F-statistic: 5.758 on 1 and 98 DF,  p-value: 0.0183
```

Note how the information in the above output matches that from the `t.test` function. In using `lm`, we get a t value in the coefficients table of `summary(lmod1)$coefficients[2,3]`, and a p-value of `summary(lmod1)$coefficients[2,4]`. We can also see the mean values for `species_1` and `species_2`, though in slightly different forms. From the `t.test` function, we see an estimated mean of 150.4545 for species 1 and 174.7862 for species 2 (this is at the bottom of the output, under `mean of x mean of y`). In the `lm`, we get the same information in a slightly different form. The estimate in the coefficients table for the intercept is listed as 150.455; this is the value of the mean height for species 1.

Where is the value for the mean height of species 2? We get the value for species 2 by adding the estimate of its effect on the line below, such that $150.455 + 24.332 = 174.786$. To understand why, think back to that `lm` structure, `plant_data$height ~ 1 + plant_data$species_ID`. Recall from Lindeløv how this is a short-hand for the familiar equation $y = \beta_0 + \beta_1 x$. In this equation, y is the dependent variable plant height, while the value x is what we might call a dummy variable. It indicates whether or not the plant in question is a member of species 2. If yes, then $x = 1$. If no, then $x = 0$.

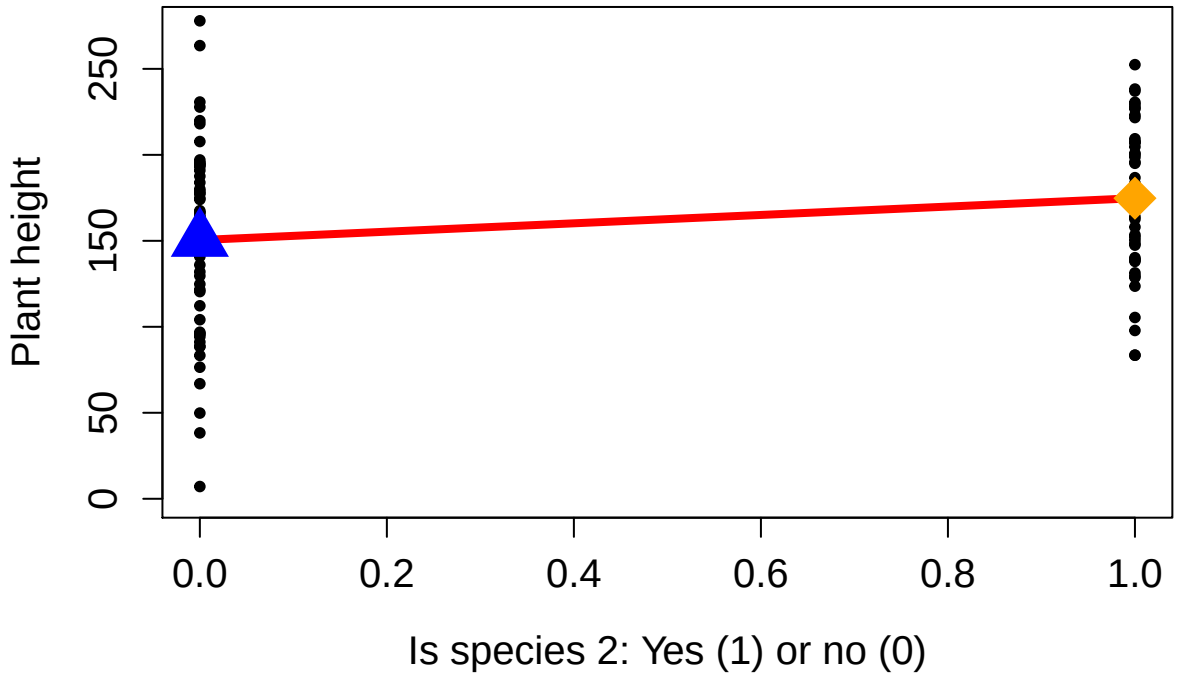
Now think about the coefficients β_0 and β_1. Because $x = 0$ whenever `species_ID = species_1`, the predicted plant height y for species 1 is simply $y = \beta_0 + (\beta_1 \times 0)$, which simplifies to $y = \beta_0$. This is why our `Estimate` of the `(Intercept)` row in the `summary(lmod1)` output equals the mean plant height of species 1. Next, because $x = 1$ whenever `species_ID = species_2`, the predicted plant height y for species 2 is $y = \beta_0 + (\beta_1 \times 1)$, which simplifies to $y = \beta_0 + \beta_1$. This is why our `Estimate` of the `plant_data$species_IDspecies_2` row in the `summary(lmod1)` equals 24.332. It is the amount that needs to be added to the prediction for species 1 to get the prediction for species 2.

To further clarify the concept, we can re-write that original two column table from above, but instead of having `species_1` or `species_2` for the `species_ID` column, we can replace it with a column that is `is_species_2`. A value of `is_species_2 = 0` means the plant is species 1, and a value of `is_species_2 = 1` means the plant is species 2.

height	is_species_2
186.72	1
237.09	1
83.51	1

height	is_species_2
121.60	0
174.04	0
183.89	0
195.81	1
153.98	0
94.22	0
162.58	1

If we now plot `is_species_2` on the x-axis, and `height` on the y-axis, we reproduce those same icons as in Lindeløv.



The blue triangle shows the mean height of species 1 (i.e., the intercept of the linear model, β_0), and the orange diamond shows the mean height of species 2 (i.e., $\beta_0 + \beta_1$). Since the distance between these two points is one, the slope of the line (rise over run) is identical to the difference between the mean species heights. Hence the reason for why β_1, which we often think about only as the 'slope' is also the difference between means.

Further equivalence of `t.test`, `lm`, and now `aov`

Analysis of variance (ANOVA) tests the null hypothesis that the mean values of groups are all equal. We often think of this being used for group numbers of three or more, but it is worth showing that ANOVA is equivalent to a t-test when the number of groups is two. A one-way ANOVA can be run using the `aov` function in R. Below, I do this for the same `plant_data` table as used for `t.test` and `lm`.

```r
aov_1 <- aov(height ~ species_ID, data = plant_data);
summary(aov_1);
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## species_ID   1  14653   14653   5.758 0.0183 *
## Residuals   98 249367    2545
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note the `F value` and the `Pr(>F)` (i.e., the p-value) in the table above. The value 5.758 matches the `F-statistic` produced from using `lm` in the previous section, and 0.0183 is the same p-value that we calculated earlier. The methods are effectively the same.

Testing for a difference between means using randomisation

An alternative approach to testing to the `t.test`, `lm`, and `aov` options above is to use randomisation. Randomisation approaches make fewer assumptions about the data, and I believe that they are often more intuitive. For a full discussion of randomisation techniques, see my previous notes for Stirling Coding Club, which goes into much more detail on the underlying logic of randomisation, bootstrap, and Monte Carlo methods. For now, I just want to illustrate how a randomisation approach can get be used for the same null hypothesis testing as shown in the previous methods above. Let us look back at the first ten rows of the data set that I made up.

height	species_ID
186.72	species_2
237.09	species_2
83.51	species_2
121.60	species_1
174.04	species_1
183.89	species_1
195.81	species_2
153.98	species_1
94.22	species_1
162.58	species_2

When we use null hypothesis testing, what we are asking is this:

> If the difference between group means is the same (null hypothesis), then what is the probability of getting a difference between groups as or more extreme than the difference that we observe in the data?

We might phrase the null hypothesis slightly differently:

> If the difference between group means is random with respect to group identity (null hypothesis), then what is the probability of getting a difference between groups as or more extreme than the difference that we observe in the data?

In other words, what if we were to randomly re-shuffle species IDs, so that we *knew* any difference between mean species heights was attributable to chance? What would the distribution of this difference look like, and where would our actual difference fall within this distribution? The logic behind randomisation here is to randomly re-shuffle group identity many times, then build a distribution for differences between randomly generated groups. We can do this with a bit of code below. First let's get the actual difference between mean heights of species 1 and species 2, i.e., `species[1] - species[2]`. We can use the `tapply` function in R to

do this easily.

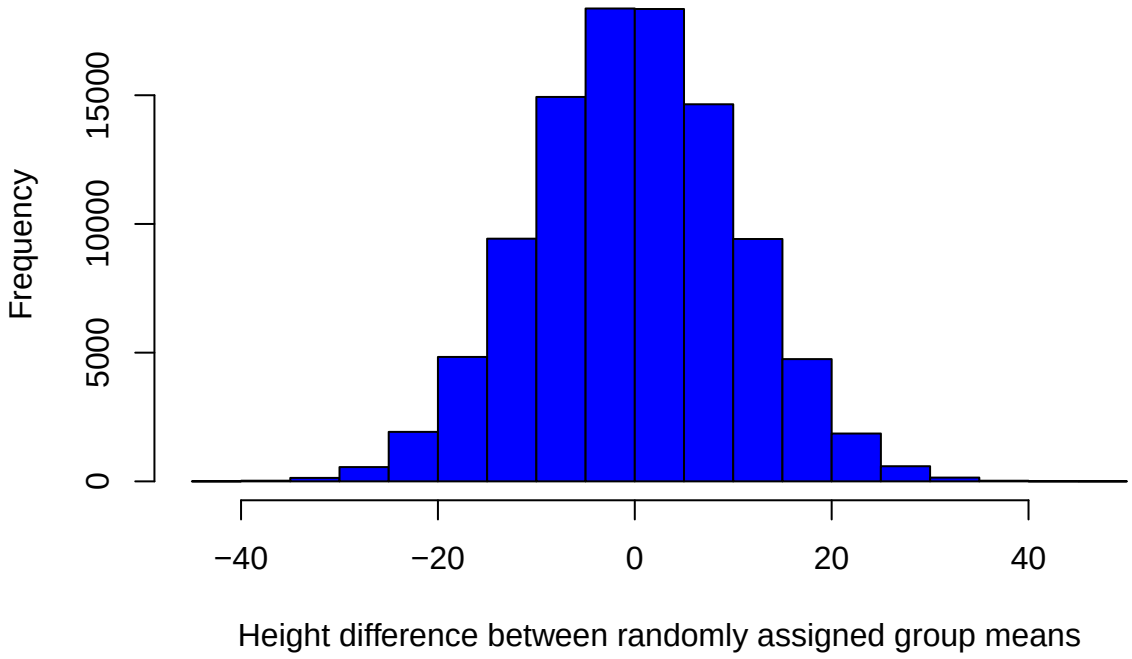
```
species <- tapply(X = plant_data$height, INDEX = plant_data$species_ID,
                  FUN = mean);
height_diffs <- as.numeric( species[1] - species[2] );
print(height_diffs);
```

```
## [1] -24.33168
```

Using a for loop in R, we can shuffle `species_ID`, then build a distribution showing what the difference between species means would be just due to random chance.

```
null_diff  <- NULL;  # Place where the random diffs will go
iterations <- 99999; # Number of reshuffles
iter       <- 1;     # Start with the first
while(iter < iterations){
  new_species_ID  <- sample(x    = plant_data$species_ID,
                            size = length(plant_data$species_ID));
  new_species     <- tapply(X = plant_data$height, INDEX = new_species_ID,
                            FUN = mean);
  new_diffs       <- as.numeric( new_species[1] - new_species[2] );
  null_diff[iter] <- new_diffs;
  iter            <- iter + 1;
}
```

Each element in `null_diff` is now a difference between the mean of species 1 and the mean of species 2, given a random shuffling of species IDs. We can look at the distribution of `null_diff` in the histogram below.



As expected, most differences between randomly assigned species height means are somewhere around zero.

Our actual value of -24.332, which we have calculated several times now, is quite low, and on the extreme tail of the distribution above. What then is the probability of getting a value this extreme if species ID has nothing to do with plant height? The answer is just the total number of values equal or more extreme to the one we observed (-24.332), divided by the total number of values that we tried (99999 + 1 = 100000; the plus one is for the actual value).

```
p_value <- sum(abs(null_diff) > abs(height_diffs)) / 100000;
```

We get `p_value` = 0.01784. Notice how close this value is to the p-value that we obtained using `t.test`, `lm`, and `aov`. This is because the concept is the same; given that the null hypothesis is true, what is the probability of getting a value as or more extreme than the one actually observed?

What about when there are more than two groups?

I want to briefly touch on what happens when there are more than three groups; for example, if we had three species instead of two. Of course, a t-test is now not applicable, but we can still use the linear model and ANOVA approaches. Let's use another data set, but with three species this time.

height	species_ID
128.83	species_3
196.58	species_1
129.02	species_3
137.57	species_1
206.67	species_1
188.68	species_3
189.37	species_3
187.66	species_1
174.81	species_3
117.31	species_1

As already mentioned, `t.test` will not work. But we can run both `lm` and `aov` with the exact same code as before with three groups. I will show `aov` first.

```
aov_2 <- aov(height ~ species_ID, data = plant_data);
summary(aov_2);
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## species_ID   2  16522    8261   5.097 0.00786 **
## Residuals   97 157212    1621
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-statistic calculated above is 5.097, and the p-value is 0.0079. The p-value in this case tests the null hypothesis that all groups (i.e., species) have the same mean values (i.e., heights). We can now use the `lm` function to run the same analysis with three groups.

```
lmod2 <- lm(height ~ 1 + species_ID, data = plant_data);
summary(lmod2);
```

```
##
## Call:
## lm(formula = height ~ 1 + species_ID, data = plant_data)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -103.33  -27.87   -2.91   28.15  115.27
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           146.839      6.904  21.268  < 2e-16 ***
## species_IDspecies_2    30.449      9.764   3.118  0.00239 **
## species_IDspecies_3     9.311      9.916   0.939  0.35007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.26 on 97 degrees of freedom
## Multiple R-squared:  0.0951, Adjusted R-squared:  0.07644
## F-statistic: 5.097 on 2 and 97 DF,  p-value: 0.007855
```

We can find the F-statistic and p-value at the very bottom of the output, and note that they are the same as reported by `aov`. But look at what is going on with the `Estimate` values in the table (ignore the `Pr(>|t|)` values in the table). There are now three rows. Again, we can think back to the equation predicting plant height y, but now we need another coefficient. The equation can now be expressed as, $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. Note that subscripts have been added to x. This is because we now have two dummy variables; is the plant species 1 (if so, $x_1 = 0$ and $x_2 = 0$), species 2 ($x_1 = 1$ and $x_2 = 0$), or species 3 ($x_1 = 0$ and $x_2 = 1$)? With these dummy variables, we can now predict the height of species 1,

$$y = \beta_0 + (\beta_1 \times 0) + (\beta_2 \times x_2).$$

The above reduces to $y = \beta_0$, as with our two species case. The height of species 2 can be predicted as below,

$$y = \beta_0 + (\beta_1 \times 1) + (\beta_2 \times x_2).$$

The above reduces to $y = \beta_0 + \beta_1$, again, as with the two species case. Finally, we can use the linear model to predict the height of species 3 plants,

$$y = \beta_0 + (\beta_1 \times 0) + (\beta_2 \times 1).$$

The above reduces to $y = \beta_0 + \beta_2$. Let's use the `tapply` function to see what the mean values of each species are in the new data set.

```
tapply(X = plant_data$height, INDEX = plant_data$species_ID, FUN = mean);
```

```
## species_1 species_2 species_3
##  146.8394  177.2879  156.1500
```

Now look at that output `summary(lmod2)` again. Notice that the estimate of the intercept `(Intercept)` is the same as the mean height of species 1 (146.8394118). Similarly, add the intercept (β_0) to the coefficient in the second row, `plant_data$species_IDspecies_2` (i.e., β_1); this value equals the mean estimate for species 2. Finally, add the intercept to the coefficient in the third row `plant_data$species_IDspecies_3` (i.e., β_2); this value equals the mean estimate for species 3. Once again, we see how the linear model is equivalent to the ANOVA.

Okay, but what's really happening with three groups?

How does this *really* work? We have given R a single column with three different categorical values (species) and somehow ended up with an intercept and two regression coefficients. How can understand this more

clearly? Think back to the table from earlier where we had a column for `is_species_2` with a simple zero or one. We can do the same, but with a new column, to include the ID of species 3.

height	is_species_2	is_species_3
128.83	0	1
196.58	0	0
129.02	0	1
137.57	0	0
206.67	0	0
188.68	0	1
189.37	0	1
187.66	0	0
174.81	0	1
117.31	0	0

In fact, for even more clarity, we can add a column for the intercept too.

height	the_intercept	is_species_2	is_species_3
128.83	1	0	1
196.58	1	0	0
129.02	1	0	1
137.57	1	0	0
206.67	1	0	0
188.68	1	0	1
189.37	1	0	1
187.66	1	0	0
174.81	1	0	1
117.31	1	0	0

Now, we can see the equivalence with the linear model expressed in R from above, `lm(height ~ 1 + species_ID, data = plant_data)`. The formula in `lm` is predicting `height` for individual plants using a linear model that includes the intercept (always 1), plus species ID. This relates now more easily to the equation $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. The four terms are reflected in the four columns above. Plant height (y) is predicted in the left-most column. The second column is all ones, by which we multipy the intercept (β_0). Columns two and three define the `species_ID` in R, and the $\beta_1 x_1 + \beta_2 x_2$ terms of the equation. Note that either $x_1 = 0$ and $x_2 = 0$ (the row is species 1), $x_1 = 1$ and $x_2 = 0$ (species 2), or $x_1 = 0$ and $x_2 = 1$ (species 3). Hence, the coefficients β_1 and β_2 apply only for species 2 and 3, repsectively (and the absence of both occurs for species 1). You should be able to connect this concept with the output of `summary(lmod2)` above.

Now if we want to predict the height of any plant (rows), we can do so just by multiplying the values in columns 2-4 (always 1 or 0) by the corresponding regression coefficients. For example, where `is_species_2 = 0` and `is_species_3 = 0`, we have $y = (\beta_0 \times 1) + (\beta_1 \times 0) + (\beta_2 \times 0)$. Substituting the regression coefficients from the `summary(lmod2)` above, we have,

$$y = (146.839 \times 1) + (30.449 \times 0) + (9.311 \times 0).$$

Note that the above simplifies to 146.839, the predicted height of species 1. We can do the same for species 2.

$$y = (146.839 \times 1) + (30.449 \times 1) + (9.311 \times 0).$$

The above simplifies to $y = 146.839 + 30.449$, which equals 177.288, the predicted height of species 2. I will leave the predicted height of species 3 to the reader.

Note that we have been working with categorical variables, species. These are represented by ones and zeroes. But we can also imagine that some other continuous variable might be included in the model. For example, perhaps the altitude at which the plant was collected is also potentially important for predicting plant height. The common name for this model would be 'ANCOVA', but all that we would really be doing is adding one more column to the table above. The column would be 'altitude', and would perhaps include values expressing metres above sea level (e.g., `altitude = 23.42, 32.49, 10.02`, and so forth; one for each plant). This value would be multiplied by a new coefficient β_3 to predict plant height, and be represented as a `lm` in R with `lm(height ~ 1 + species_ID + altitude, data = plant_data)`. Its equation would be $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$, where altitude is x_3.

Can we make this even more elegant, somehow?

I hope that all of this has further illustrated some of the mathematics and code underlying linear models. Readers who are satisfied can skip this section, but I want to go just one step further and demonstrate how linear model prediction really boils down to **just one equation**. This equation is a generalisation of the by now familiar $y = \beta_0 + \beta_1 x$. We can represent independent and dependent variables in the table above using columns of two matrices, Y and X. Y is just a vector of 100 plant heights (matching column 1 from the table above),

$$Y = \begin{pmatrix} 128.83 \\ 196.58 \\ 129.02 \\ 137.57 \\ \vdots \\ 232.97 \end{pmatrix}.$$

Similarly, X is just a matrix with 100 rows and 3 columns indicating the intercept and species identities, as in columns 2-4 above,

$$X = \begin{pmatrix} 1, & 0, & 1 \\ 1, & 0, & 0 \\ 1, & 0, & 1 \\ 1, & 0, & 0 \\ \vdots & & \\ 1, & 1, & 0 \end{pmatrix}.$$

What we want to figure out now are the coefficients for predicting values in Y (i.e., matrix elements) from values in each of the columns of X. In other words, what are the values of β_0, β_1, β_2, which were explained in the last section? We can also represent these values in a matrix β,

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

We have already figured out what these values are using `lm` in R. They are just the `Estimate` values from the output of `summary(lmod2)` in an earlier section. But we can also predict them using a bit of matrix algebra, solving for β given the values in Y and X. The generalisation of $y = \beta_0 + \beta_1 x$ is the compact equation below,

$$Y = X\beta.$$

For our data, we could therefore substute for Y and X matrices,

$$\begin{pmatrix} 128.83 \\ 196.58 \\ 129.02 \\ 137.57 \\ \vdots \\ 232.97 \end{pmatrix} = \begin{pmatrix} 1, & 0, & 1 \\ 1, & 0, & 0 \\ 1, & 0, & 1 \\ 1, & 0, & 0 \\ \vdots & & \\ 1, & 1, & 0 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

Given this equation, we now only need to solve for β to get our coefficients predicting Y from X. This requires some knowledge of matrix multiplication and matrix inversion. These topics are a lesson in themselves, so I will not go into any detail as to what these matrix operations do. The point is that we are trying to isolate β in the equation $Y = X\beta$. My hope is that a rough idea of what is going on is possible even for those unfamiliar with matrix algebra, but feel free to skip ahead.

Isolating β with matrix algebra

I should note that all of the matrix algebra that you have seen here is thanks to Dean Adams at Iowa State University (but if you notice any errors, they are mine, not his). The first thing that we need to isolate β is to multiply both sides of the equation $Y = \beta X$ by the transpose of X, X^t,

$$X^tY = X^tX\beta.$$

Next, we multiply both sides by the inverse of (X^tX), $(X^tX)^{-1}$,

$$\left(X^tX\right)^{-1} X^tY = \left(X^tX\right)^{-1} X^tX\beta.$$

Notice now that on the right side of the equation, we have $\left(X^tX\right)^{-1} X^tX$. In other words, we multiply the inverse of (X^tX) by itself, thereby cancelling itself out (getting the identity matrix, the matrix algebra equivalent of 1). That leaves us only with β on the right hand side of the equation, which is exactly what we want. We can flip this around and put β on the left side of the equation,

$$\beta = \left(X^tX\right)^{-1} X^tY.$$

Hence, we have isolated β, and can use the right side of the above equation (where the data are located) to get our predictors.

Using one equation to get predictions of coefficients.

We now have our equation for getting our prediction coefficients β,

$$\beta = \left(X^tX\right)^{-1} X^tY.$$

Now I will use this equations to rederive our regression coefficients. **Do not worry about the details here**. What I want to show is that the above equation really does get us the same regression coefficents that we got from the output of `summary(lmod2)` earlier. Here is that output again.

```
##
## Call:
## lm(formula = height ~ 1 + species_ID, data = plant_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -103.33  -27.87   -2.91   28.15  115.27
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          146.839      6.904  21.268  < 2e-16 ***
## species_IDspecies_2   30.449      9.764   3.118  0.00239 **
## species_IDspecies_3    9.311      9.916   0.939  0.35007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.26 on 97 degrees of freedom
## Multiple R-squared:  0.0951, Adjusted R-squared:  0.07644
## F-statistic: 5.097 on 2 and 97 DF,  p-value: 0.007855
```

Now let's use the data in `plant_data` to calculate β manually instead. Here are the first ten rows of `plant_data` again.

height	the_intercept	is_species_2	is_species_3
128.83	1	0	1
196.58	1	0	0
129.02	1	0	1
137.57	1	0	0
206.67	1	0	0
188.68	1	0	1
189.37	1	0	1
187.66	1	0	0
174.81	1	0	1
117.31	1	0	0

We want to set the first column as a matrix Y, and the remaining columns as a matrix X.

```
Y <- as.matrix(plant_data[,1]);
X <- as.matrix(plant_data[,2:4]);
```

Here are the first five elements of `Y`.

```
## [1] 128.83 196.58 129.02 137.57 206.67
```

Here are the first five rows of `X`.

```
##      the_intercept is_species_2 is_species_3
## [1,]             1            0            1
## [2,]             1            0            0
## [3,]             1            0            1
## [4,]             1            0            0
## [5,]             1            0            0
```

Let's do the matrix algebra now below to get values of β. Note that in R, matrix multiplication is denoted by the operation `%*%`. Matrix inversion of `X` is written as `solve(X)`, and matrix transpose of `X` is written as `t(X)`. Our expression of $\beta = (X^t X)^{-1} X^t Y$ in R is therefore as follows.

```
betas <- solve( t(X) %*% X ) %*% t(X) %*% Y;
```

We can now print `betas` to reveal our coefficients, just as they were reported by `lm`.

```
print(betas);
```

```
##                     [,1]
## the_intercept 146.839412
## is_species_2   30.448529
## is_species_3    9.310588
```

We have just produced our regression coefficients manually, with matrix algebra.

Some final thoughts

The reason that I have gone through this step by step is to build on our earlier exploration of common statistical tests as linear models. All linear models can be expressed using this common framework. In the above matrix example, note that we could have added as many columns as we wished to X. Perhaps we also collected data on the altitude at which we found the plants in our hypothetical example. We could add this information in as an additional column of numbers in X, then calculated a new β_4 in exactly the same way. In our new model, this would result in a discrete group predictor (species, in our case), and a continuous variable (altitude). The associated statistical test would commonly be called an ANCOVA, but all that would really have happened is that we would be adding a new column to the list of independent variables. **As an exercise**, think about how we would add interaction terms in X.

But wait, there is more. There is no reason why Y needs to be represented by a single column. Maybe we want to predict not just plant height, but plant seed production too. In other words, perhaps we have more than one dependent variable and we need a multivariate approach (e.g., MANOVA). The same equation for getting β works here too; we can think of multivariate linear models in the exact same way as univariate models; we are just adding more columns to Y.

I hope that this has been a useful supplement to the already very useful introduction by Lindeløv. There are details that I have left out for the sake of time, but my goal has been to further simplify the logic and mathematics underying linear models in statistics.

This document is entirely reproducible. Because the data are simulated, if you Knit it in Rstudio, you will get different numbers each time. I encourage you to try this, and explore the code for yourself. I have cheated in a few places just to avoid making a simulated data set that is too extreme, by chance. All of the code is posted below with some notes.

Key bits of code underlying the simulated data

The code below can be used to generate a CSV file with simulated data as shown here. Note that you can change the significance of different regression coefficients, and their magnitudes and signs, by changing how `height` is defined within the `while` loops below.

```
# The code below creates plant heights with an intercept of roughly 150
# and a beta_1 coefficient of roughly 20, with some error added into it.
# The while loop just does this to avoid any non-significant results or
# very highly significant results that arise due to chance.
species_n <- c("species_1", "species_2");
sim_pval  <- 0;
while(sim_pval > 0.05 | sim_pval < 0.001){
    species_eg  <- sample(x = species_n, size = 100, replace = TRUE);
```

```
    species1    <- as.numeric(species_eg == "species_1");
    species2    <- as.numeric(species_eg == "species_2");
    error       <- rnorm(n = 100, mean = 0, sd = 40);
    height      <- round(150 + (species2 * 20) + error, digits = 2);
    species_ID  <- as.factor(species_eg);
    plant_data  <- data.frame(height, species_ID);
    sim_mod     <- lm(plant_data$height ~ 1 + plant_data$species_ID);
    sim_pval    <- summary(sim_mod)$coefficients[2,4];
}
write.csv(plant_data, file = "two_discrete_x_values.csv", row.names = FALSE);

# Below does the same job as above, just with three species instead of two
species_n <- c("species_1", "species_2", "species_3");
sim_pval  <- 0;
while(sim_pval > 0.05 | sim_pval < 0.001){
    species_eg  <- sample(x = species_n, size = 100, replace = TRUE);
    species1    <- as.numeric(species_eg == "species_1");
    species2    <- as.numeric(species_eg == "species_2");
    species3    <- as.numeric(species_eg == "species_3");
    error       <- rnorm(n = 100, mean = 0, sd = 40);
    height      <- round(150 + (species2 * 20) + error, digits = 2);
    species_ID  <- as.factor(species_eg);
    plant_data  <- data.frame(height, species_ID);
    sim_mod     <- lm(plant_data$height ~ 1 + plant_data$species_ID);
    sim_pval    <- summary(sim_mod)$coefficients[2,4];
}
write.csv(plant_data, file = "three_discrete_x_values.csv", row.names = FALSE);
```