**Problem**

The case\_when() function in dplyr is great for dealing with multiple complex conditions (if’s). But how do you specify an “else” condition in case\_when()?

**Context**

Last month, [I was super excited to discover the case\_when() function](https://thewoodpeckr.wordpress.com/2019/10/11/if-ifelse-had-more-ifs/) in dplyr. But when I showed my blog post to a friend, he pointed out a problem: there seemed to be no way to specify a “background” case, like the “else” in ifelse(). In the previous post, I gave an example with three outcomes based on test results. The implication was that there would be roughly equal numbers of people in each group. But what if the vast majority of people failed both tests, and we really just wanted to filter out the ones who didn’t?

Today, I came across exactly this problem in my research. I’m analyzing morphometric data for about 500 tadpoles, and I made a PCA score plot that looked like this:

Before continuing my analysis, I wanted to take a closer look at those outlier points, to make sure they represent real measurements and not mistakes in the data. Specifically, I wanted to take a look at these ones:

To figure out which tadpoles to investigate, I’d have to pull out their names based on their scores on the PC1 and PC2 axes.

**Solution**

I decided to add a column called investigate to the PCA scores data frame, set to “investigate” or “ok” depending on whether the observation in question needed to be looked at.

scores <- scores %>%

mutate(investigate = case\_when(PC1 > 0.2 ~ "investigate",

PC2 > 0.15 ~ "investigate",

PC1 < -0.1 & PC2 > 0.1 ~ "investigate,

TRUE ~ "ok"))

What’s up with that weird TRUE ~ "ok" line at the end of the ​​case\_when() statement? Basically, that’s the equivalent of else.  It translates, roughly, to “assign anything that’s left to “ok.”

I’m really not sure why the equivalent of else here is TRUE, and the ​case\_when [documentation](https://dplyr.tidyverse.org/reference/case_when.html) doesn’t really explain it. The only way I figured out that this worked was by reading through the examples in the documentation and noticing that they all seemed to end with this TRUE ~ statement, so I tried it, and voilà. If anyone has an understanding of why this works, under the hood, I’d love to know!

One thing to note is that the order of arguments matters here. If we had started off with the TRUE ~ "ok" statement and *then* specified the other conditions, it wouldn’t have worked: everything would just get assigned to “ok.”

I’m really glad I figured out how to add an else to case\_when()! Before I started using dplyr, I would have attempted this problem like this:

scores$investigate <- "ok" # Create a whole column filled with "ok"

scores$investigate[scores$PC1 > 0.2] <- "investigate"

scores$investigate[scores$PC2 > 0.15] <- "investigate"

scores$investigate[scores$PC1 < -0.1 & scores$PC2 > 0.1] <- "investigate"

Or maybe I would have used some really long and complex boolean statement to get all those conditions in one line of code. Or nested ifelse‘s. But that’s annoying and hard to read. This is so much neater, and saves typing!

**Outcome**

It turns out that if you read the documentation closely, case\_when()is a fully-functioning version of ifelse that allows for multiple if statements AND a background condition (else). The more I learn about the tidyverse, the more I love it.