light.r: Explanation

Goals of code:

- Create indicator variables to use in a regression
- Automatically create indicator variables

**Creating an indicator variable "by hand"**: `(light$time=='E')+0`
One quick way to create an indicator variable with 0 / 1 values is to combine a logical operation and an arithmetic operation. The `==` operator is the logical equals; does the right hand side equal the left hand side? This has the values "TRUE" when time is "E" and "FALSE" when it is anything else. Internal to R, "TRUE" is stored as a 1 and "FALSE" is stored as a 0. Adding 0 to the logical value converts the logical value to a numeric value, so Itime is 1 for the "E" group and 0 for the "L" group. This is demonstrated by joining the original value, the logical, and the numeric value and printing them all (the `data.frame` call).

The Itime indicator that we just created has the class / book definition: the last level ("L") gets the 0 value.

**Using indicator values in a multiple regression**:
Nothing new. Just include the variable on the left hand side of the formula.

**Creating a interaction variable**: `light$Itime*light$intensity`
Interaction variables are created by multiplying the continuous variable by the indicator.

**Creating indicator variables automatically**: `factor(time)`
When a variable is defined as a factor, R creates indicator variables automatically. The R default is that the FIRST level gets the 0 value. (Note: this can be changed. Ask me if you want to see how). You can create the factor version of the variable in the data frame: `light$time.f <- factor(light$time)` then include `time.f` in the `lm()` model, or you can create the factor "on the fly": `light.lm1b <- lm(flowers ~ factor(time) + intensity, data=light)`. I prefer to create a new variable because it simplifies predicting new observations.

If you look at the output from `summary()`, you see a line for `time.fL`. This is the name of the factor concatenated with the level of the indicator that had the value of 1.

Although the parameter estimates depend on the definition of the indicator, the predicted values are the same (at least to machine precision). The `round( stuff )` calculates the difference between the predictions from "our" indicator and the R indicator, rounds that to 8 decimal places and prints the results. All 0. I rounded because otherwise the numbers are reported as something X $10^{-14}$ (somethingE-14 in scientific notation).

**Viewing the values for an R created indicator**: `model.matrix()`
After fitting a model, you can view the matrix of all X variables used for that fit by `model.matrix()`. The argument is the name of an `lm()` fit. This is especially helpful when there are factors in the model, because `model.matrix()` will return columns for each indicator variable. If there are no factors, the result from `model.matrix()` is just the model variables from the original data frame.

**Creating an interaction automatically**: `time.f:intensity`
R can create interaction variables automatically. The syntax is the two variable names separated by a : without any spaces.

Because models with many variables may have many interactions, R provides a short-hand for "variables and their interaction". That is `time.f*intensity`, which R expands into `time.f intensity time.f:intensity`, i.e., the two variables and their interaction.