### 5. Understanding the final model better

##### Model coefficients

coef(mod4)

* Identity effects (p1\_ID, p2\_ID, p3\_ID, p4\_ID) represent the inherent contribution of each species in a mixture. They also represent the monoculture performances of the respective species.
* The bfg\_G\_L term represents the interaction potential of mixtures containing both grasses and legumes while wfg\_G and wfg\_L represent the interaction potential between the two grasses and two legumes, respectively.
* Finally, the respective interaction terms with treatment represent the change in the contribution of the grasses (p1:treatFlowN and p2:treatFlowN) and between FG interaction (bfg\_G\_L:treatFlowN) for communities that received low nitrogen fertilisation.

##### Alternative formulations

These models are all exactly the same as the our best final model, just parametrised in a different way

###### Including the intercept in the model

$$y =β\_{0}+β\_{1}P\_{1}+β\_{2}P\_{2}+β\_{3}P\_{3}+β\_{4}P\_{4}+ω\_{GL}[(P\_{1}+P\_{2})(P\_{3}+P\_{4})] +ω\_{G}(P\_{1}P\_{2}) +ω\_{L}(P\_{3}P\_{4}) +β\_{1}^{'}P\_{1}X\_{N} +β\_{2}^{'}P\_{2}X\_{N}+ω\_{GL}^{'}\left[\left(P\_{1}+P\_{2}\right)\left(P\_{3}+P\_{4}\right)\right]X\_{N} +ϵ $$

# lm requires you to specify the full formula for model
# the formula is structured response ~ predictors
lm\_mod <- lm(# response (yield in this case)
 yield ~
 # Species identity effects
 p1 + p2 + p3 + p4 +
 # FG interaction effects
 bfg\_G\_L + wfg\_G + wfg\_L +
 # Interaction with N treatment
 p1:treatF + p2:treatF + bfg\_G\_L:treatF,
 # data used for model fitting
 data = model\_data)
summary(lm\_mod)

This model has lowN as reference (notice the reversed signs for some coefficients compared to mod4), but we can force it to have regN as reference to get a similar model to the one generated by the DI() function.

lm\_mod\_low <- lm(yield ~
 p1 + p2 + p3 + p4 +
 bfg\_G\_L + wfg\_G + wfg\_L +
 p1:treatF + p2:treatF + bfg\_G\_L:treatF,
 # Trick to force regN to be the reference
 data = model\_data %>%
 mutate(treatF = relevel(treatF, ref = "regN")))
summary(lm\_mod\_low)

###### Formatting model coefficients to show absolute values of response across treatment levels

# lm requires you to specify the full formula for model
# the formula is structured response ~ predictors
lm\_alt <- lm(yield ~ 0 + # 0 + for suppressing the intercept
 # Interested in absolute values of terms instead of difference
 p1:treatF + p2:treatF + p3 + p4 +
 bfg\_G\_L:treatF + wfg\_G + wfg\_L,
 data = model\_data)
summary(lm\_alt)

###### Dummy variable encoding for categorical variables (this will most useful for your datasets)

Dummy variable encoding (also called 0-1 or one-hot encoding) is a way of representing categorical variables by creating binary (0 or 1) columns for each level of the variable. One level is chosen as the reference (or baseline), and the remaining levels are represented as separate columns with values of 0 or 1 to indicate the presence or absence of that level for each observation. Thus, if a categorical variable has $n$ levels, $n-1$ columns are created. For our example, since treatF has two levels, one new column would be added.

Generally, the modelling function (lm, glm, DI, nlme, etc.) performs dummy variable encoding automatically under the hood when fitting the models. However, each function has a different approach to choosing the reference level which could cause confusion. Manually creating dummy variables helps avoid these problems.

# Create a dummy variable for treatment
model\_data <- model\_data %>%
 # regN will be 1 if treatF equal "regN" and 0 otherwise
 mutate("regN" = ifelse(treatF == "regN", 1, 0))

# The extra code is just to clean the output
head(model\_data %>% select(treatF, regN, everything()))

We can fit the model now just like before

mod\_final <- DI(y = "yield", prop = c("p1", "p2", "p3", "p4"),
 DImodel = "FG", FG = c("G", "G", "L", "L"),
 # Notice we use regN instead of treatF
 extra\_formula = ~ (p1 + p2 + bfg\_G\_L):regN,
 data = model\_data)

summary(mod\_final)