### 2. Model selection using autoDI()

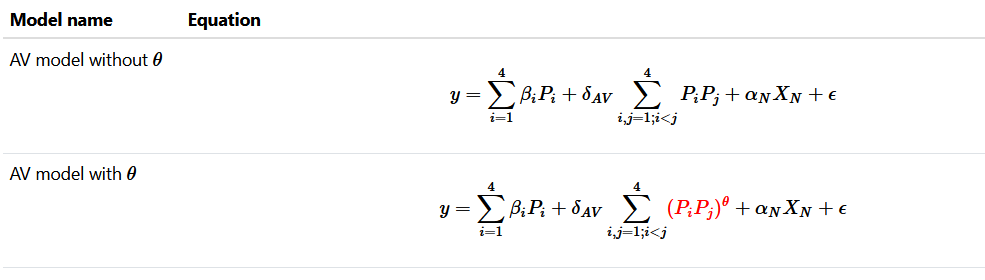
#### Code for running autoDI()

# Since treatment is a categorical variable. Convert it to a factor before modelling  
# This is dplyr pipeline which is very useful for manipulating dataframe  
# '%>%' is called the pipe operator and simply passes the object on the left  
# side to a function on the right  
model\_data <- model\_data %>%   
 # The mutate function is used to add new columns to a data-frame  
 # We add a new column called "treatF" which is a factor  
 mutate("treatF" = factor(treatment))  
# The code might look daunting but is very handy for complex data manipulation  
# operations as we will see later.  
# The above code is equivalent to running the following  
# model\_data$treatF <- factor(model\_data$treatment)  
  
auto\_mod <- autoDI(# y specifies the name of the column containing response  
 y = "yield",   
 # prop accepts the names of columns containing species proportions   
 prop = c("p1", "p2", "p3", "p4"),   
 # the name of column containing treatments (if present) goes in treat  
 treat = "treatF",  
 # FG accepts a character string describing functional grouping of species  
 FG = c("G", "G", "L", "L"),  
 # step0 tests against an intercept only model  
 step0 = TRUE,  
 # Finally, data accepts name of data-frame with these columns  
 data = model\_data)

#### Equations of models fit by autoDI() at each step

##### Step0: Investigating non-diversity experimental structures

##### Step 1: Investigating



##### Step 2: Investigating interaction structures

##### Step 3: Investigating effect of treatment

##### Step 4: Assessing lack of fit

##### Best model selected by autoDI()

summary(auto\_mod)

##### Predicting for a 50:50 grass mixture

##### Predictions for specific communities using model selected by autoDI()

# Extract the four monocultures, 50:50 grass and legume mixtures and   
# centroid community from data  
subset\_comms <- model\_data %>%   
 # Filter the specific communities by their number in the dataset  
 filter(community %in% c(1, 2, 3, 4, 5, 6, 27)) %>%   
 # Remove any duplicates  
 distinct(p1, p2, p3, p4)  
  
subset\_comms  
  
# Use the prediction\_contributions function from DImodelsVis package  
prediction\_contributions(# model accepts a DImodel object  
 model = auto\_mod,   
 # data accepts a data-frame with species communities to show on plot  
 data = subset\_comms,  
 # bar\_labs can be used to give labels to the X-axis  
 bar\_labs = c("G1", "G2", "L1", "L2", "GG", "LL", "GGLL"))