### 7. Interpreting the final model using visualisations

##### Visualising species contributions to predicted response

# Extract the four monocultures, and centroid community from data at two N levels
subset\_comms <- model\_data %>%
 # Filter the specific communities by their number in the dataset
 filter(community %in% c(1, 2, 3, 4, 27)) %>%
 # Remove any duplicates and only include columns used in the model
 distinct(p1, p2, p3, p4, bfg\_G\_L, regN)

subset\_comms

# Use the prediction\_contributions function from DImodelsVis package
prediction\_contributions(# model accepts a DImodel object
 model = mod\_final,
 # 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", "G1\_reg", "G2", "G2\_reg", "L1", "L1\_reg", "L2", "L2\_reg", "Cent", "Cent\_reg"))

# Try different sets of communities in the data and try to create a
# similar plot for all equiproportional communities in the original data

##### Average change in yield over richness

# This function can be used to visualise the predicted response for
# specific communities shown using pie-glyphs and show the average
# change in response over the richness gradient
gradient\_change(# DI model object
 model = mod\_final,
 # data to be used for making predictions
 data = model\_data,
 # Specify any treatment variables in here as list
 # A separate plot will be created for each value specified here
 add\_var = list(regN = c(0, 1)),
 # How to arrange the created plots in a grid
 nrow = 1)

# Try different datasets in data and see how the average relationship changes
# Hint generate average relationship for richness using all possible
# equi-proportional communities by specifying the following in data argument
# `get\_equi\_comms(4, variables = c("p1", "p2", "p3", "p4"))`

##### Visualise effects of adding or removing a species from a community

# This function allows to visualise the changes in the predicted response
# if a particular species is added or removed from a community whilst
# keeping the relative proportions of the remaining species constant
# Visualising these changes for several communities could show the average
# effect of adding/removing any species

# First we need to choose the communities that we wish to visualise
# We will visualise the effect of adding each of four species to all the
# communities in the original design at lowN level
lowN\_data <- model\_data %>%
 # filter only those communities receiving low N
 filter(regN == 0) %>%
 # Select only the species proportions and dummy variable
 select(p1, p2, p3, p4, regN)
head(lowN\_data)

visualise\_effects(# DImodel object
 model = mod\_final,
 # Data containing communities to be visualised
 data = lowN\_data,
 # Which species to focus on, will choose all by default
 # But remove a particular species from here if needed
 var\_interest = c("p1", "p2", "p3", "p4"),
 # Visualising the effect of adding a species
 # change to "decrease" to visualise effect of species loss
 effect = "increase",
 # Whether to show confidence interval around prediction
 se = FALSE,
 # Whether to show the average effect
 average = TRUE)

# In this example we will visualise the effect of adding each of four
# species to all the communities in the original design at regN level
regN\_data <- model\_data %>%
 # filter only those communities receiving regular N
 filter(regN == 1) %>%
 # Select only the species proportions and dummy variable
 select(p1, p2, p3, p4, regN)
head(regN\_data)

visualise\_effects(# DImodel object
 model = mod\_final,
 # Data containing communities to be visualised
 data = regN\_data,
 # Which species to focus on, will choose all by default
 # But remove a particular species from here if needed
 var\_interest = c("p1", "p2", "p3", "p4"),
 # Visualising the effect of adding a species
 # change to "decrease" to visualise effect of species loss
 effect = "increase",
 # Whether to show confidence interval around prediction
 se = FALSE,
 # Whether to show the average effect
 average = TRUE)

# Visualise the above two plots by adjusting the aesthetics of the plot
# or specifying different communities in the data, perhaps those which
# weren't already present in the design. Also visualise the uncertainty.

##### Visualise the predicted response across the simplex space

# Visualise the predicted response by changing the proportion of three
# species while fixing the proportion of remaining species to be constant

# The response would be shown as a contour map in a ternary diagram with
# one species fixed to be at value P while the remaining three to vary
# between 0 and 1 - P.

# We will prepare the data first
cond\_tern\_data <- conditional\_ternary\_data(
 # column name of species
 prop = c("p1", "p2", "p3", "p4"),
 # Models object
 model = mod\_final,
 # Which three species to show within the ternary
 tern\_vars = c("p1", "p2", "p3"),
 # Name of the fourth species along with fixed values
 conditional = data.frame("p4" = c(0, 0.25, 0.5)),
 # Same as before, any additional treatment variables
 # One plot for each value specified here
 add\_var = list(regN = c(0, 1)),
 # increase this to make the plot more detailed
 # but higher values would imply longer runtimes
 resolution = 1,
 # Predictions can be made directly but holding off
 # for now as we need to add the between\_FG term
 prediction = FALSE)
# See structure of data
head(cond\_tern\_data)

# Add the bfg\_G\_L column to the data
cond\_tern\_data <- cond\_tern\_data %>%
 # Same as before, using DI\_data and appending FG columns to data
 bind\_cols(DI\_data(prop = c("p1", "p2", "p3", "p4"),
 what = "FG",
 FG = c("G", "G", "L", "L"),
 data = .))

# Add predictions now
cond\_tern\_data <- cond\_tern\_data %>%
 # Helper function to add predictions
 add\_prediction(model = mod\_final)
# This is how final data looks
head(cond\_tern\_data)

conditional\_ternary\_plot(# Data generated before with predictions
 data = cond\_tern\_data,
 # don't print numbers on contours,
 contour\_text = FALSE,
 # lower limit to show on legend
 lower\_lim = 6,
 # upper limit to show on legend
 upper\_lim = 18,
 # number of levels for contour
 nlevels = 8,
 # Arrange resultant plots in two rows
 nrow = 2)

# Try more examples by adapting the above code and trying different values
# for p4 or trying to fix a different species to be constant

##### Visualise the response at the functional group level

# This function can be used to visualise the predicted response at the
# functional group level by combining species within a functional group.
# The resultant plot is shown as a ternary diagram

# Same as conditional ternary we prepare the data first
group\_tern\_data <- grouped\_ternary\_data(
 # column name of species
 prop = c("p1", "p2", "p3", "p4"),
 # Models object
 model = mod\_final,
 # Which species to group in a functional group
 # There should be three unique groups specified here
 # In this example, we group the legumes together but leave
 # the grasses ungrouped.
 # Alternatively we could group grasses as follows
 # FG = c("G", "G", "L1", "L2")
 FG = c("G1", "G2", "L", "L"),
 # The ratio in which the FG proportion should be split
 # between the component species
 # G1 and G2 are in their own groups so are 100% each while
 # the total legume proportion is split 50-50 between
 # p3 and p4. This can be changed however, for example,
 # an 80-20 split can be specified as c(1, 1, 0.8, 0.2)
 values = c(1, 1, 0.5, 0.5),
 # Additional treatment variables
 add\_var = list(regN = c(0, 1)),
 # increase this to make the plot more detailed
 # but higher values would imply longer runtimes
 resolution = 1
)

# See structure of data
head(group\_tern\_data)

# Add the bfg\_G\_L column to the data
group\_tern\_data <- group\_tern\_data %>%
 # Same as before, using DI\_data and appending FG columns to data
 bind\_cols(DI\_data(prop = c("p1", "p2", "p3", "p4"),
 what = "FG",
 FG = c("G", "G", "L", "L"),
 data = .))

# Add predictions now
group\_tern\_data <- group\_tern\_data %>%
 # Helper function to add predictions
 add\_prediction(model = mod\_final)
# This is how final data looks
head(group\_tern\_data)

# All parameters are same as before
grouped\_ternary\_plot(# Data generated before with predictions
 data = group\_tern\_data,
 # don't print numbers on contours,
 contour\_text = FALSE,
 # lower limit to show on legend
 lower\_lim = 6,
 # upper limit to show on legend
 upper\_lim = 18,
 # number of levels for contour
 nlevels = 8,
 # Arrange resultant plots in one rows
 nrow = 1)

# Try more examples of this visualisation by trying different ratios of
# splitting the total legume proportion or grouping the grasses instead.