Workflow: LUI and species coexistence

Rodrigo R. Granjel

11/02/2020

1 Prepare the data

First of all, we need to re-structure the database. We will only use data from 2008 to 2015. For this period, we select the 26 most common plant species for further analyses. The reason why we select a 26 species community is because, at this moment, it is impossible to compute the analyses for a richer community (e.g. 50 species) due to computation limitations. To give an example, computing all posible combinations of 11 species in a 50 species community would take ca. 220 years, while computing this for a 26 species community takes ca. 17 days (there are ca. 4835 times more possible combinations for the 50 species community).

1.1 Land Use Intensity (LUI)

First of all, we define the path to load the raw data:

```
#in the end, raw data needs to go to a public repository
raw_path <- "data/raw_data"
```

And then we start by the LUI dataset:

```
#load LUI data
lui <- read.csv(paste(raw_path, "LUI06_15.csv", sep = "/"), header  = TRUE)

#stick with LUI at different years
lui.only <- lui[, grep("LUI", names(lui))]
lui.only2 <- lui.only[, -c(1:2)] ## remove 2006 and 2007
lui_tot <- cbind("plot" = lui$Plot, lui.only2)

#restructure LUI
lu <- c(lui_tot$LUI_08, lui_tot$LUI_09, lui_tot$LUI_10, lui_tot$LUI_11,
        lui_tot$LUI_12, lui_tot$LUI_13, lui_tot$LUI_14, lui_tot$LUI_15)
yy <- c(rep(2008, nrow(lui_tot)), rep(2009, nrow(lui_tot)),
        rep(2010, nrow(lui_tot)), rep(2011, nrow(lui_tot)),
        rep(2012, nrow(lui_tot)), rep(2013, nrow(lui_tot)),
        rep(2014, nrow(lui_tot)), rep(2015, nrow(lui_tot)))
pp <- rep(lui_tot$plot, length(unique(yy)))

#create a re-structured LUI dataset
lui_total <- data.frame("Plot" = pp, "Year" = yy, "LUI" = lu)

#clean
rm(lui, lui_tot, lui.only, lui.only2, pp, lu, yy)
```

1.2 Plant species data

1.2.1 Prepare quantitative data

And continue with the plants dataset:

In this case, though, we will save two different data files, both of which we will merge with the previously obtained LUI data. The first one that we obtain corresponds to the original data for the 26 most common species in the different plots, with a final column representing the cumulative data for all other species present at each plot, which we will call `rest`. Note that we also re-scale the data to avoid having values over 100%. **We will use this to quantify the intra- and inter-specific interaction coefficients for all plant species.**

```
#this chunk depends on chunk(s): restructure_lui

#number of plants
n_plants <- 26

#load plant data
plants <- read.csv(paste(raw_path, "BE.plants08.16.csv", sep = "/"), header = TRUE)

#remove data from 2016
plants <- plants[plants$Year != 2016, ]

#remove info columns
#format is the same order as LUI, so we'll later merge with LUI to have all info
plants <- plants[-c(1:5)]

### top --- select the most common plant species
top <- rev(sort(apply(plants[, -c(1:5)], 2, mean, na.rm = TRUE)))[1:n_plants]

#short standard names for the selected plants --- WARNING: CURRENTLY, A MANUAL STEP
top.short <- c("Poa_tri", "Poa_pra", "Alo_pra", "Dac_glo", "Tri_rep", "Tar_off",
               "Lol_per", "Arr_ela", "Fes_rub", "Fes_pra", "Tri_fla", "Ely_rep",
               "Tri_pra", "Ran_rep", "Bro_ere", "Ran_acr", "Bro_hor", "Pla_lan",
               "Ant_syl", "Her_sph", "Gal_mol", "Hol_lan", "Hel_pub", "Car_hir",
               "Bra_pin", "Pha_aru")

#the following species would be: "Ant_odo", "Ver_cha", "Fes_ovi", "Rum_ace", "Des_ces",
#"Phl_pra", "Agr_sto", "Cyn_cri", "Cir_ole", "Cre_bie", "Cer_hol", "Pla_med", "Thy_pul",
#"Urt_dio", "Lol_mul", "Cir_aru", "Ran_bul", "Tri_dub", "Lot_cor", "Car_car", "Leo_his",
#"Vic_sep","Med_lup", "Pru_spp", "Sym_off")

#sum all the other plant species per row
rest <- apply(plants[, -match(names(top), names(plants))], 1, sum, na.rm = TRUE)

#dataset with the most common plant species
plants <- plants[, match(names(top), names(plants))]

#give them standard short names
names(plants) <- top.short

#add a column with the sum of the non-selected plant species
q_plants <- cbind(plants, rest); rm(rest)
```

```r
# to re-scale to no more than 100%
q_plants <- q_plants / apply(q_plants, 1, sum) #q_plants = quantitative plant data

#add plot, year & LUI information
q_plants <- cbind(lui_total, q_plants)

#remove NAs
q_plants <- na.omit(q_plants)

#save q_plants
write.table(q_plants, "data/q_plants.txt", row.names = FALSE, sep = "\t")
```

1.2.2 Prepare qualitative data (presence/absence)

The second one corresponds to the presence or absence of the 26 most common plant species. To do so, we change the value of each cell for a 1 (values over 0) or a 0 (values equal to zero) where corresponds. This will allow us to explore the relationship between species richness and LUI and also to select the feasible combinations for each richness level. (The following code uses the data frame `plants`, obtained in the previous code chunk.)

```r
#this chunk depends on chunk(s): restructure_lui, restructure_plants_quanti

#change to presence/absence data (0 or 1)
for (i in 1:nrow(plants)){
  for (j in 1:ncol(plants)){
    if(is.na(plants[i, j])){
      plants[i, j] <- 0
    } else {
      if (plants[i, j] > 0){
        plants[i, j] <- 1
      }
    }
  }
}

#final data frame for plant presence/absence
p_plants <- cbind(lui_total, plants)

#remove NAs
p_plants <- na.omit(p_plants)

#write p_plants
write.table(p_plants, "data/p_plants.txt", row.names = FALSE, sep = "\t")

#clean a bit
rm(plants, i, j)
```

2 Data description for decision-making

2.1 Range of LUI and relationship with species richness

Now, we want to do two things:

- Undestand the range of our natural variability in LUI.
- Describe the relationship between the LUI and species richness, both for a community with all the species in our data (full community) and our 26-species community.

To do so, we already have the LUI data in good shape (`lui_total`), so we start by preparing the dataset for the full plant community:

```
#this chunk depends on chunk(s): restructure_lui

#load plant data
plants <- read.csv(paste(raw_path, "BE.plants08.16.csv", sep = "/"), header = TRUE)

#remove data from 2016
plants <- plants[plants$Year != 2016, ]

#remove info columns
#format is the same order as LUI, so we'll later merge with LUI to have all info
plants <- plants[-c(1:5)]

#change to presence/absence data (0 or 1)
for (i in 1:nrow(plants)){
  for (j in 5:ncol(plants)){
    if(is.na(plants[i, j])){
      plants[i, j] <- 0
    } else {
      if (plants[i, j] > 0){
        plants[i, j] <- 1
      }
    }
  }
}

#number of species per plot
rich_full <- apply(plants[, -c(1:5)], 1, sum, na.rm = TRUE)

#merge lui and richness
richness_full_comm <- cbind(lui_total, rich_full)

#year as factor for plotting
richness_full_comm$Year <- as.factor(richness_full_comm$Year)

#save full community presence data
write.table(richness_full_comm, "data/richness_full_comm.txt",
            row.names = FALSE, sep = "\t")

#clean
rm(i, j, rich_full)
```

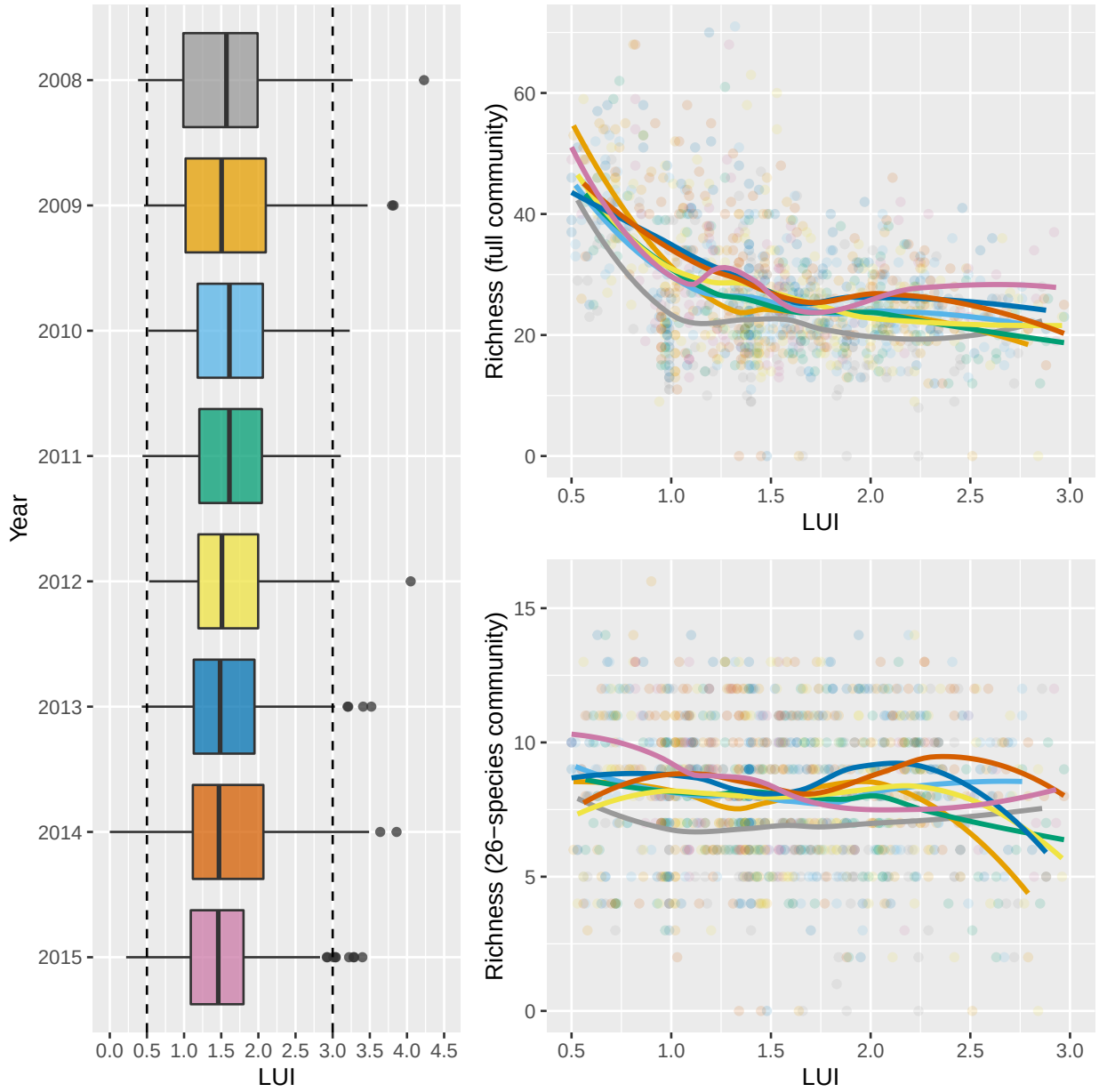
Then, we apply an equivalent procedure to obtain the richness by datapoint of our 26-species community:

```
#this chunk depends on chunk(s): restructure_lui, lui_richness_full_comm

#dataset with the most common plant species
plants <- plants[, match(names(top), names(plants))]

#give them standard short names
names(plants) <- top.short

#number of species per plot
rich_26spp <- apply(plants[, -c(1:5)], 1, sum, na.rm = TRUE)

#merge lui and richness
richness_26spp_comm <- cbind(lui_total, rich_26spp)

#year as factor for plotting
richness_26spp_comm$Year <- as.factor(richness_26spp_comm$Year)

#save full community presence data
write.table(richness_26spp_comm, "data/richness_26spp_comm.txt",
            row.names = FALSE, sep = "\t")

#clean
rm(lui_total, plants, rich_26spp)
```

And, finally, with both data frames ready we can proceed to plot the data to describe the variability of LUI and the relationship between LUI and species richness for both the full and 26-species community.

```
#load ggplot2
library(ggplot2)

#color-blind friendly colour palette - do not lose clean this, it's very useful
color.blind <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
                 "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

#boxplot of LUI through the years
box_lui <- ggplot(data = richness_full_comm, aes(x = Year, y = LUI)) +
  geom_boxplot(aes(fill = Year), alpha = 0.75) +
  theme(legend.position = "none") +
  geom_abline(intercept = 3, slope = 0, linetype = "dashed") +
  geom_abline(intercept = 0.5, slope = 0, linetype = "dashed") +
  scale_x_discrete(limits = rev(levels(richness_full_comm$Year))) +
  scale_y_continuous(limits = c(0, 4.5), breaks = seq(0, 4.5, by = 0.5)) +
  coord_flip() +
  scale_fill_manual(values = color.blind)

#plot of the relationship between LUI and richness in different years (full community)
lui_rich_full <- ggplot(data = richness_full_comm,
                        aes(x = LUI, y = rich_full, colour = Year)) +
  geom_point(alpha = 0.15) + geom_smooth(alpha = 0, size = 1.1) +
  scale_x_continuous(limits = c(0.5, 3), breaks = seq(0.5, 3, by = 0.5)) +
  scale_y_continuous(position = "left") + ylab("Richness (full community)") +
  theme(legend.position = "none") + scale_colour_manual(values = color.blind)

#plot of the relationship between LUI and richness in different years (26-species comm)
```

```
lui_rich_26spp <- ggplot(data = richness_26spp_comm,
                         aes(x = LUI, y = rich_26spp, colour = Year)) +
  geom_point(alpha = 0.15) + geom_smooth(alpha = 0, size = 1.1) +
  scale_x_continuous(limits = c(0.5, 3), breaks = seq(0.5, 3, by = 0.5)) +
  scale_y_continuous(position = "left") +
  ylab("Richness (26-species community)") +
  theme(legend.position = "none") +
  scale_colour_manual(values = color.blind)

#load the ggpubr library for charts (nicely combining plots)
library(ggpubr)

#step one: combine the full community and 26-spp plots
both_lui_rich <- ggarrange(lui_rich_full, lui_rich_26spp, nrow = 2, align = "hv")

#step two: combine all
lui_richness <- ggarrange(box_lui, both_lui_rich, widths = c(3, 4))
print(lui_richness)
```




```
#save the figure as png (change output format as you please)
ggsave(plot = lui_richness, "figures/LUI_richness.png",
       width = 7, height = 7, dpi = 320)

#clean everything
rm(list=ls()[! ls() %in% c("n_plants")])
```

In brief, **we observe that our LUI values concentrate between 0.5 and 3.0, so we will not analyse what happens beyond this range.**

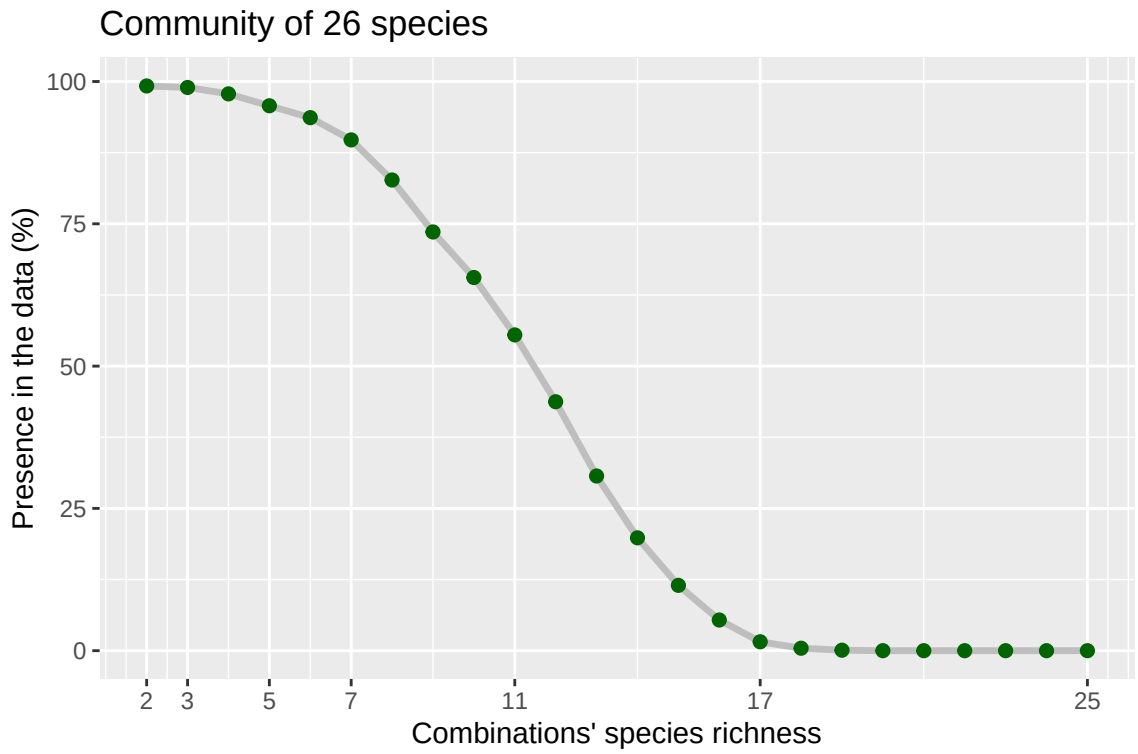
2.2 Feasible combinations for each richness level

Picking which combinations we are going to use to apply the structural coexistence framework is not trivial. Given that the structural coexistence framework is extremely computation-demanding, it is imperative to carefully select the interactions to compute and remove those that don't make sense, biologically speaking (e.g, combinations that are theoretically feasible but can't be found in nature: our data).

2.2.1 Select the richness levels for the analyses

First of all, we ought to understand how well represented are the different combinations of different richness levels in our data. This way, we avoid choosing a richness level that has actually no representation in nature.

```
#load qualitative plants
p_plants <- read.table("data/p_plants.txt", header = TRUE, sep = "\t")

#data only with the plants in columns
p_plants_only <- p_plants[, 4:(n_plants + 3)]

#calculate the percentage of presence of different richness values in the data
max_comb <- NULL
rich <- NULL
for (i in 2:25){
  rich <- c(rich, i)
  max_comb <- c(max_comb,
                length(which(apply(p_plants_only, 1, sum) > i))
                / nrow(p_plants_only) * 100)
}

#save as dataframe for plotting
saturation <- data.frame("richness" = rich, "ratio" = max_comb)

#plot `saturation`
(ratio_combos <- ggplot(data = saturation, aes(x = richness, y = ratio)) +
    scale_x_continuous(breaks = c(2, 3, 5, 7, 11, 17, 25)) +
    ggtitle("Community of 26 species") +
    xlab("Combinations' species richness") +
    ylab("Presence in the data (%)") +
    geom_line(size = 1.2, color = "grey") +
    geom_point(size = 2, color = "darkgreen"))
```




```
#save the plot
ggsave(plot = ratio_combos, filename = "figures/ratio_combos.png",
       width = 6, height = 4, dpi = 320)

#clean (except for p_plants_only, that we'll use below)
rm(i, max_comb, rich, ratio_combos, p_plants, saturation)
```

As can be observed, **the plot above urges us to select combinations that do not involve more than 17 plant species.**

2.2.2 Pick the combinations for each richness level

Now, we need to select the specific combinations that are feasible (i.e., the ones that are actually present in our data). To do so, we have created a function called `selector` that, given a data frame —with only plants in columns— and a richness value (ranging from 2 to `ncol(df) - 1`), provides us with a data frame of combinations above a certain `threshold` of presence of that given combination in the data. For instance, if our data has 10 entries and a certain combination of species is present in 2 spots, that combinations will be selected by the function if we set a threshold below 20%.

The function looks as follows:

```
#function to select the combinations above a certain threshold of presence
selector <- function(df, n, threshold){

  #vector with the names of all the species
  vec <- colnames(df)

  #define a vector with the maximum values different elements can reach
  maxim <- vec[(length(vec) - (n - 1)):length(vec)]
```

```r
#data frame to save the selected combinations
sel_comb <- data.frame()

#vector to save the combination to look for at each moment
combo <- vec[1:n]

#counter of times the combo is present
counter <- 0

#starting with the first combo
for (w in 1:nrow(df)){

  #if all the species of the combo are present at a given spot
  if(isFALSE(0 %in% (df[w, which((vec %in% combo) == TRUE)]))){
    counter <- counter + 1 #add one to te counter
  }
}

#if the presence overcomes a certain threshold, in %, the combos is saved
if ((counter / w * 100) > threshold){
  sel_comb <- rbind(sel_comb, cbind(t(combo), (counter / w * 100)))
}


#loop i starting at row two for all the rows
i = 2 #define the starting point
while (i != choose(length(vec), n)){

  #loop j for the elements of the combo vector, starting at position 2
  for (j in 1:n){

    #conditions to match
    if (combo[j] != maxim[j]){ #while the value is not the maximum

      if (j != n){ #if the position is not the last one yet
        #do nothing, continue to the next element

      } else { #if the element is, in fact, the last one
        combo[j] <- vec[which(combo[j] == vec) + 1] #write the following corresp. value
      }

    } else { #if it is actually the maximum value for that element
      #write the following corresponding values from the previous position to the end
      combo[(j - 1):n] <- vec[(which(combo[j - 1] == vec) + 1):
                                ((which(combo[j - 1] == vec) + 1) + ((n - j) + 1))]
      break #enough - changes have been made
    }
  } #end j

  #counter of times the combo is present
  counter <- 0

  #continuing with all the other combos
```

```
    for (w in 1:nrow(df)){

      #if all the species of the combo are present at a given spot
      if(isFALSE(0 %in%(df[w, which((vec %in% combo) == TRUE)]))){
        counter <- counter + 1 #add one to te counter
      }
    }

    #if the presence overcomes a certain threshold, in %, the combos is saved
    if ((counter / w * 100) > threshold){
      sel_comb <- rbind(sel_comb, cbind(t(combo), (counter / w * 100)))
    }

    #next step
    i <- i + 1

  } #end i while
  return(sel_comb) #return the final df with all saved combinations
} #end selector()
```

And then we need to compute the function for different richness levels. This is an important step. Given that we may choose richness levels from 2 to 17, **we have decided to use the following sequence: 2, 3, 5, 7, 11 and 17 species**. In this sequence, each element is the result of the sum of the two previous elements minus one (e.g., `11 = (5 + 7) - 1`). Besides, the following piece of code should be run in a cluster of computation because it takes a long time to finish —that's why the code is commented (annuled by `#`). In any case, note that we use the `p_plants_only` dataset and a threshold value of zero (if the combination is present at any given spot, we save it).

```
##set a threshold (in %)
#t <- 0
#
##compute and save the selected combos (TO DO IN THE CLUSTER OF COMPUTATION!)
#res2 <- selector(p_plants_only, n = 2, threshold = t)
#res3 <- selector(p_plants_only, n = 3, threshold = t)
#res5 <- selector(p_plants_only, n = 5, threshold = t)
#res7 <- selector(p_plants_only, n = 7, threshold = t)
#res11 <- selector(p_plants_only, n = 11, threshold = t)
#res17 <- selector(p_plants_only, n = 17, threshold = t)

#clean a bit
rm(selector, p_plants_only)
```

After running the code in the cluster, we may load the combos selected for each richness level:

```
#load combos of 2 species
combos2 <- read.table("results/selected_combos/combos2.txt", sep = "\t", header = TRUE)

#load combos of 3 species
combos3 <- read.table("results/selected_combos/combos3.txt", sep = "\t", header = TRUE)

#load combos of 5 species
combos5 <- read.table("results/selected_combos/combos5.txt", sep = "\t", header = TRUE)

##NOT READY YET --- load combos of 7 species
#combos7 <- read.table("results/selected_combos/combos7.txt", sep = "\t", header = TRUE)
```

```r
##NOT READY YET --- load combos of 11 species
#combos11 <- read.table("results/selected_combos/combos11.txt", sep = "\t", header = TRUE)

#load combos of 17 species
combos17 <- read.table("results/selected_combos/combos17.txt", sep = "\t", header = TRUE)
```