Temporal dynamics of seed-dispersal networks: Disentangling the role of direct and indirect biotic and climatic processes

Andrés F. Ramírez-Mejía, Corey E. Tarwater, J. Patrick Kelley, Jinelle H. Sperry, Jeffrey T. Foster, Donald R. Drake, and Jeferson Vizentin-Bugoni

Appendix S4. Journal: Ecology

Table of contents

S1 Introduction

This document holds data wrangling operations and procedures for estimating the interaction functional space and network metrics from **Ramírez-Mejía et al. 'Temporal dynamics of**

seed-dispersal networks: Disentangling the role of direct and indirect biotic and climatic processes'. The final sections provides the code used for joining climatic, network metrics and functional space data in a single relational dataset for fitting the final models.

S2 Overall data wrangling

The following code conduct operations for cleaning data and reviewing compatibility of species names and labels across data sets

```r
sapply(c('dplyr', 'ggplot2', 'lubridate', 'forcats',
         'magrittr', 'cmdstanr', 'rethinking', 'cowplot',
         'bipartite', 'igraph', 'iNEXT', 'hypervolume', 'readxl'),
       library, character.only = T)

extrafont::loadfonts(device = 'win')

source('functions_mod_diagnostics.r')
```

```r
path_alien_native <- '/Users/andres/Documents/github_repos/hawaii_projects/fecal_songbirds/native_plants_alien.xl

alien_native1 <- read_xlsx(path_alien_native, sheet = 1, col_names = T)
alien_native2 <- read_xlsx(path_alien_native, sheet = 2, col_names = T)
alien_native2$native <- ifelse(alien_native2$native == 'A', F, T)

alien_native <-
  full_join(alien_native1,
            alien_native2,
            by = c('sp_plant', 'native')) |> unique()

df <- readRDS('FecalSongbirds_CORRECTED.rds')

codes_birds <- readRDS('birds_code.rds')

codes_plants <- readRDS('plants_code.rds')
codes_plants2 <- read_xlsx('plants_codes2.xlsx', sheet = 1, col_names = T)

plant_traits <- readRDS('plant_traits.rds')

bird_traits <- readRDS('bird_traits.rds')

df <- df[, c("Site", "Sample_Date", "Month", "Bird_sp",
             "PlantSpecies", "SeedCount")]

colnames(df) <- c('site', 'date', 'month', 'bird', 'plant', 'seeds')

df <- df[df$plant != 'none', ]

df$code <- df %$% paste0(bird, '_', plant)

df <- df[!grepl('[0-9]', df$code), ]

df <- df[!is.na(df$seeds), ]

df$plant <- toupper(df$plant)

codes_plants$code <- toupper(codes_plants$code)

missing_plants <- unique(df$plant)[!(unique(df$plant) %in% unique(codes_plants2$Acronym))]
```

```r
plants_codes_network <- codes_plants2[codes_plants2$Acronym %in% unique(df$plant), ]

codes_plants$species <-
  gsub('^(.*)(\\s)([a-z]*)(\\s)(.*)$', '\\1\\2\\3', codes_plants$species)

colnames(plants_codes_network) <- c('species', 'code', 'origin')

plants_codes_full <- full_join(codes_plants, plants_codes_network, by = c('code', 'species'))

plants_codes_full  <- plants_codes_full[, 1:4]

plants_codes_full$genus <- gsub('^(.*)(\\s)(.*)$', '\\1', plants_codes_full$species)

codes_temp <-
  cbind(
  plants_codes_full[which(is.na(plants_codes_full$family)), c(1:2, 5)],
  tribble(~family,       ~order,
          'Urticaceae',  'Rosales',
          'Araliaceae',  'Apiales',
          'Moraceae',     'Rosales',
          'Moraceae',     'Rosales',
          'Moraceae',     'Rosales',
          'Verbenaceae',  'Lamiales',
          'Euphorbiaceae', 'Malpighiales',
          'Passifloaraceae', 'Malpighiales',
          'Dipentodontaceae', 'Huerteales',
          'Araliaceae', 'Apiales',
          'Boraginaceae', 'Boraginales',
          'Euphorbiaceae', 'Malpighiales',
          'Cucurbitaceae', 'Cucurbitales')
) |> as_tibble()

plants_codes_full <-
  full_join(plants_codes_full[-which(is.na(plants_codes_full$family)), ],
            codes_temp, by = c('code', 'species', 'genus', 'family', 'order'))
plants_codes_full <-
  rbind(plants_codes_full,
      tibble(code = missing_plants,
             species = NA,
             family = 'Arecaceae',
             order = 'Arecales',
             genus = NA))

names(plant_traits)[!(names(plant_traits) %in% plants_codes_full$species)] <- 'Sambucus nigra'

plant_traits$`Sambucus nigra`$sp <- 'Sambucus nigra'

codes_birds$Acronym <- toupper(codes_birds$Acronym)
```

S3 Sampling completeness

Rarefaction analysis to evaluate the completeness of sampled interactions among at each site.

```r
interactions <- unique(df$code)

rarefaction_dat <- split(df, df$site)

rarefaction_dat <-
  lapply(rarefaction_dat, FUN =
           function(x) {

             sapply(interactions, FUN =
                      function(i) {
                        length(which(x$code == i))
                      })

           })

rarefaction_plot <- iNEXT(rarefaction_dat, q = 0, datatype = 'abundance')
```

```
ggiNEXT(rarefaction_plot) +
  theme_minimal() +
  labs(x = 'Interaction events', y = 'Interaction richness')
```

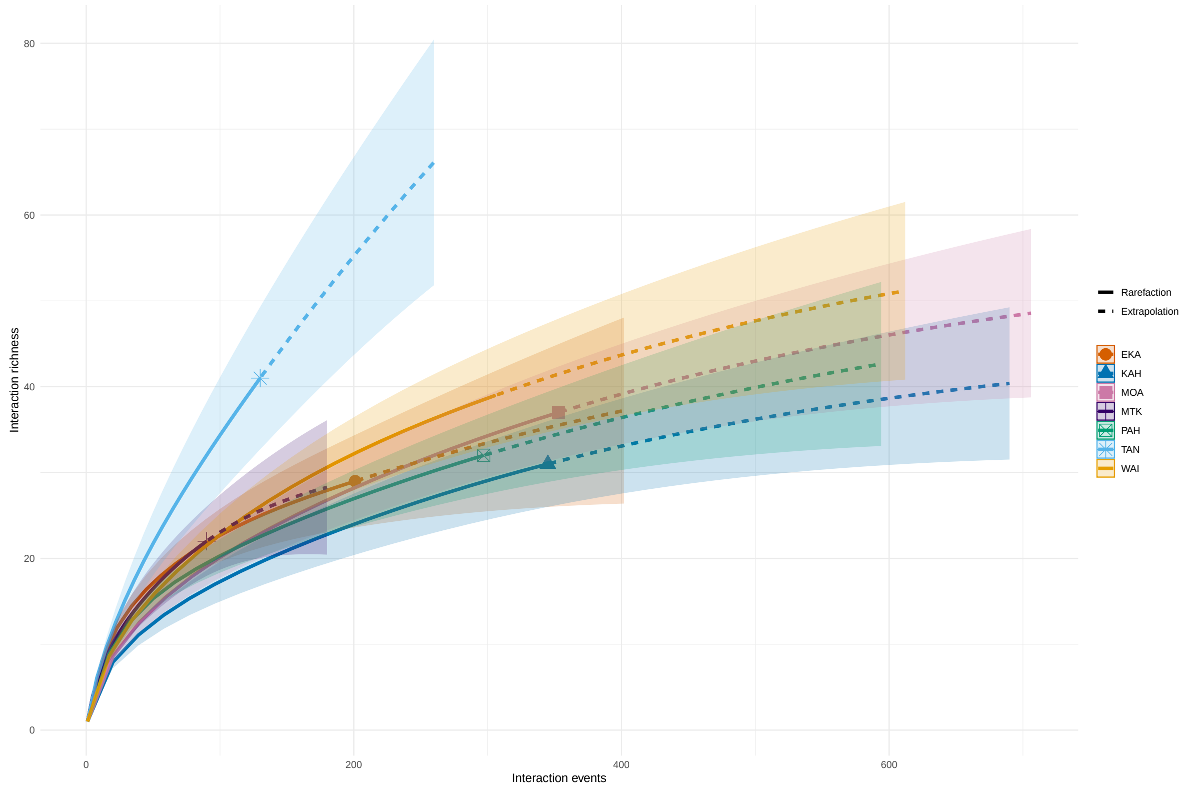

Figure S1: Rarefaction curves of seed dispersal interactions in seven sites at the Oahu island, Hawaii

S4 Network metrics

S4.1 Data wrangling

Data wrangling tasks for exploring interactions, plotting networks and estimating its metrics

```r
df <- split(df, list(df$site, df$month))

df <- df[unlist(lapply(df, function(x) nrow(x) > 0), use.names = F)]

par(mfrow = c(1, 2))
plot(density(unlist(lapply(df, function(x) length(unique(x$bird))), use.names = T)),
     main = '', xlab = 'Bird species per network')
plot(density(unlist(lapply(df, function(x) length(unique(x$plant))), use.names = T)),
     main = '', xlab = 'plant species per network')
```

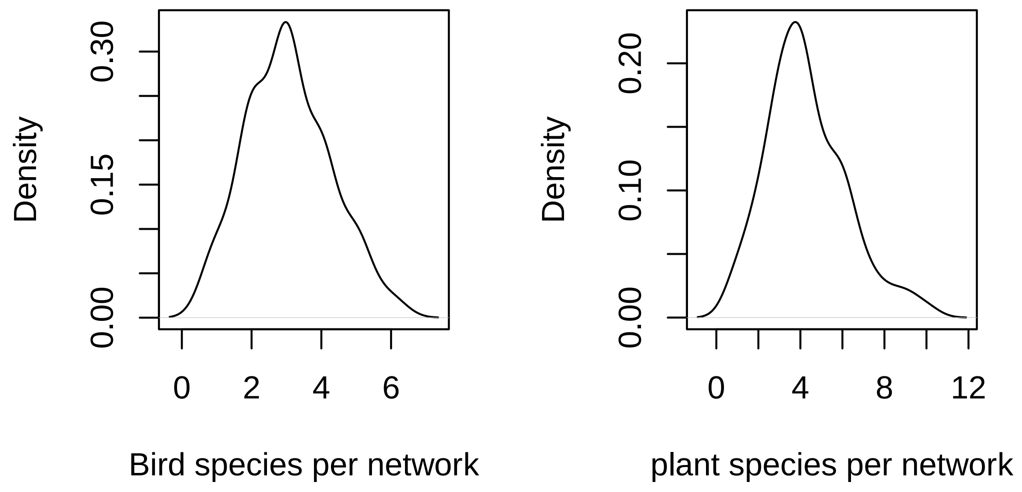

Figure S2: Distribution of richness of plant and bird species integrating seed dispersal networks

```r
par(mfrow = c(1, 1))
```

```r
N_networks <-
  do.call('rbind',
          lapply(df, FUN =
                   function(x) {
                     tibble(N_birds = length(unique(x$bird)),
                            N_plants = length(unique(x$plant)),
                            site = x$site[1],
                            month = x$month[1],
                            N = length(unique(x$date))) # N sampling
                   }))

final_nets_indx <-
  lapply(df, FUN =
         function(x) {
           i <- length(unique(x$bird))
           j <- length(unique(x$plant))

           i >= 2 & j >=2
         }) |> unlist()

df <- df[final_nets_indx]

matrix_networks <-
  lapply(df, FUN =
           function(x) {
             x$code <- toupper(x$code)
             d_temp <- unique(x[, c('bird', 'plant')])
             d_temp <- as.matrix(d_temp)
             d_temp <- expand.grid(d_temp[,1], d_temp[, 2])
             d_temp <- as_tibble(unique(d_temp))
             colnames(d_temp) <- c('bird', 'plant')
             d_temp$code <- d_temp %$% paste(bird, plant, sep = '_')
             d_temp <- full_join(x, d_temp, by = c('bird', 'plant', 'code'))
             indx <- which(is.na(d_temp$seeds))
             d_temp$seeds[indx] <- 0
             d_temp$site[indx] <- d_temp$site[1]
             d_temp$month[indx] <- d_temp$month[1]
             d_temp$year <- year(d_temp$date)

             d_temp <-
               d_temp |>
               unique() |>
               group_by(year, code) |>
               mutate(seeds = sum(seeds)) |>
               unique() |>
               ungroup() |>
               group_by(code) |>
               mutate(seeds = mean(seeds)) |>
               ungroup() |>
               dplyr::select(site, month, code, bird,
                              plant, seeds) |>
               unique() |>
               mutate(freq_rel = seeds/sum(seeds))
             d_temp <- d_temp[order(d_temp$seeds, decreasing = T), ]

             birds <- unique(d_temp$bird)
             plants <- unique(d_temp$plant)

             sapply(plants, FUN =
                      function(i) {
                        sapply(birds, FUN =
                                 function(j) {
                                   d_temp[d_temp$bird == j &
                                            d_temp$plant == i,]$freq_rel
                                 })
                      })

           })


matrix_networks_plots <-
  lapply(matrix_networks, FUN =
           function(x) {
             w <- t(x)
             w <- w[which(w > 0)]
             #net <- graph_from_biadjacency_matrix(x)
             net <- graph_from_incidence_matrix(x) # to be run in mac
             V(net)$size <- igraph::degree(net) * 5
             V(net)$label <- ''
             V(net)$color <- c(rep('seagreen', nrow(x)), rep('tan1', ncol(x)))
             E(net)$weight <- w
             net
           })
```

```
Warning: `graph_from_incidence_matrix()` was deprecated in igraph 1.6.0.
i Please use `graph_from_biadjacency_matrix()` instead.
```

S4.2 Network metrics plotting

```
par(mfrow = c(8, 4), mar = c(1, 1, 1, 1))
for (i in seq_along(matrix_networks)) {
  plot(matrix_networks_plots[[i]],
       layout = layout_in_circle,
       edge.width = E(matrix_networks_plots[[i]])$weight * 10)
  text(x = 0, y = 1.1, lab = names(matrix_networks_plots)[i])
}
```

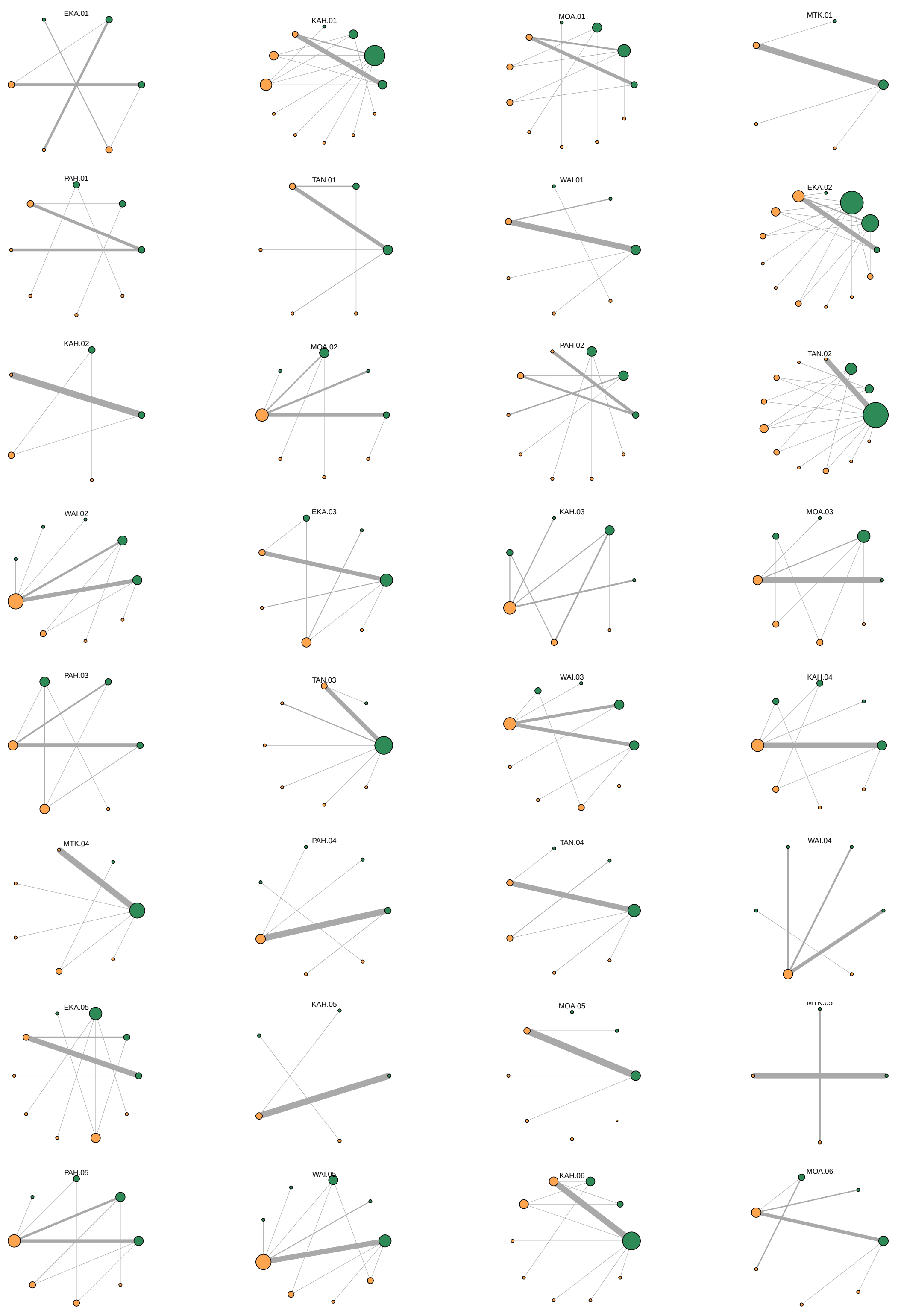

Figure S3: Seed dispersal networks per site and month (top labels) at the Oahu island. Green: plants; orange: birds. The size of the circles are weighted by the degree of the node

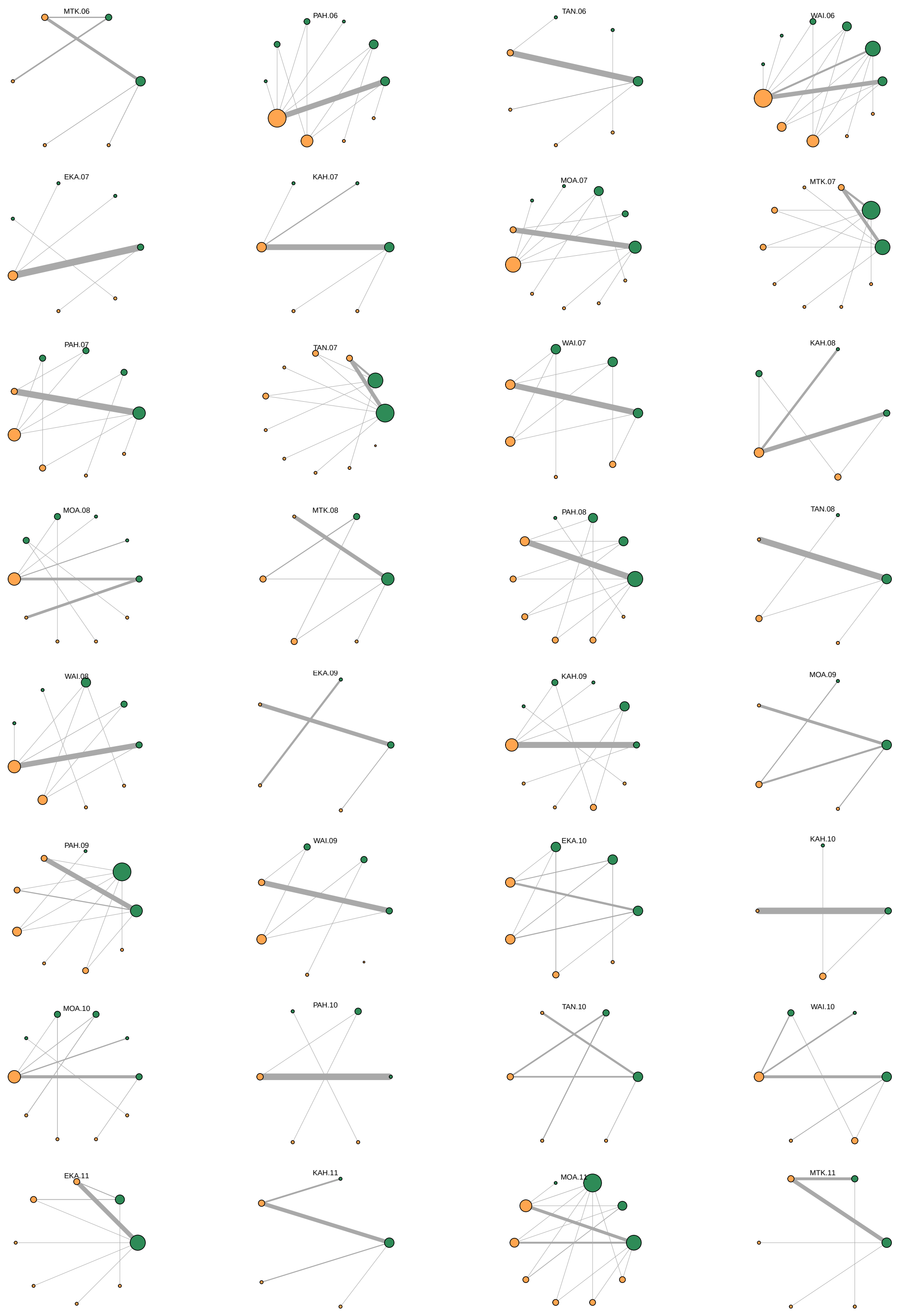

Figure S4: Seed dispersal networks per site and month (top labels) at the Oahu island. Green: plants; orange: birds. The size of the circles are weighted by the degree of the node

```r
par(mfrow = c(1, 1))
```

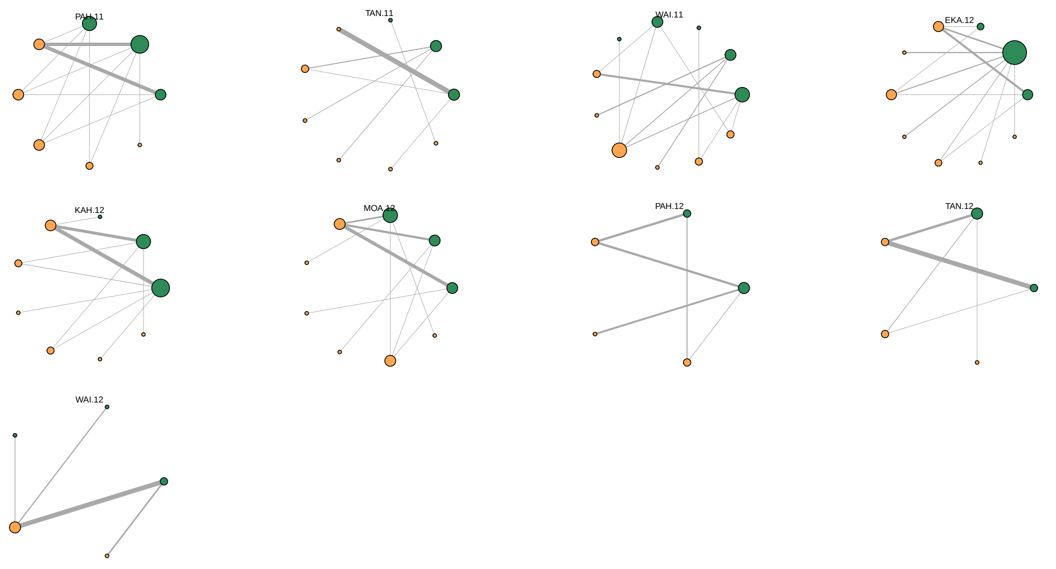

Figure S5: Seed dispersal networks per site and month (top labels) at the Oahu island. Green: plants; orange: birds. The size of the circles are weighted by the degree of the node

```
metrics_networks <-
  do.call('rbind',
          lapply(seq_along(matrix_networks), FUN =
                   function(x) {
                     t <- matrix_networks[[x]]

                     modu <- computeModules(t)

                     tibble(size = nrow(t) + ncol(t),
                            N_birds = nrow(t),
                            N_plants = ncol(t),
                            asimetry = (N_birds - N_plants) / (N_plants + N_birds),
                            nestedness = networklevel(t, index = 'weighted NODF'),
                            modularity = modu@likelihood,
                            H2 = networklevel(t, index = 'H2'),
                            site = names(matrix_networks)[x])
                   }))
```

S4.3 Network metrics ∼ network size

```
par(mfrow = c(2, 2), mar = c(4, 4, 1, 1))
for (i in 1:4) {
  plot(metrics_networks[[i]], metrics_networks$nestedness,
       xlab = colnames(metrics_networks)[i], ylab = 'nestedness')
}
```

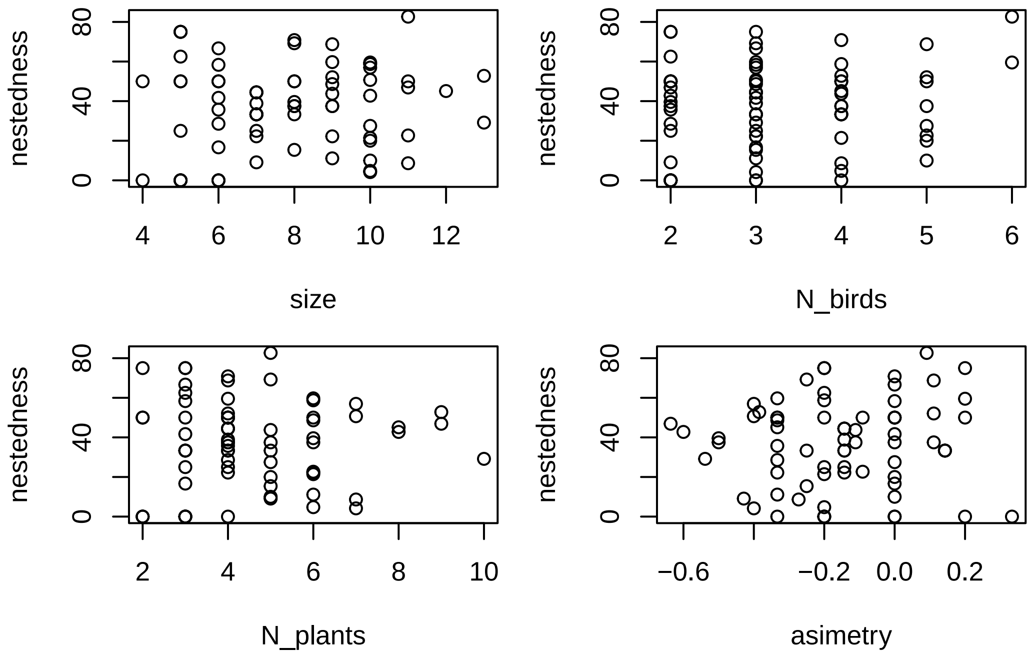

Figure S6: Scatter plots showing relationships between network metrics (y-axis), and their size (i.e., total species, N birds, N plants, asymetry)

```
for (i in 1:4) {
  plot(metrics_networks[[i]], metrics_networks$H2,
       xlab = colnames(metrics_networks)[i], ylab = 'H2')
}
```

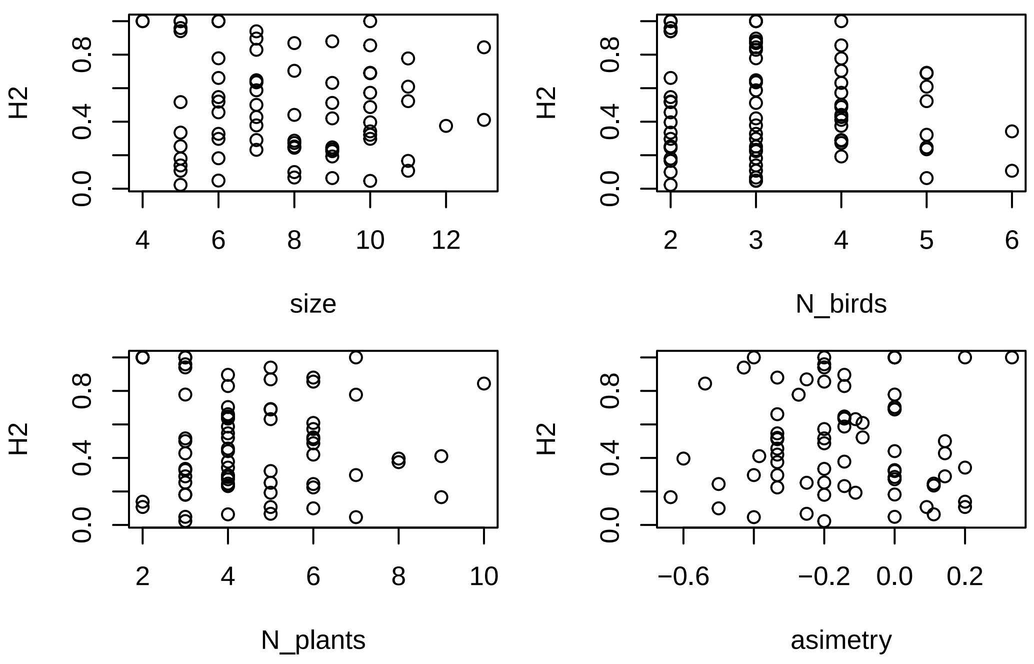

Figure S7: Scatter plots showing relationships between network metrics (y-axis), and their size (i.e., total species, N birds, N plants, asymetry)

```
for (i in 1:4) {
  plot(metrics_networks[[i]], metrics_networks$modularity,
       xlab = colnames(metrics_networks)[i], ylab = 'Modularity')
}
```

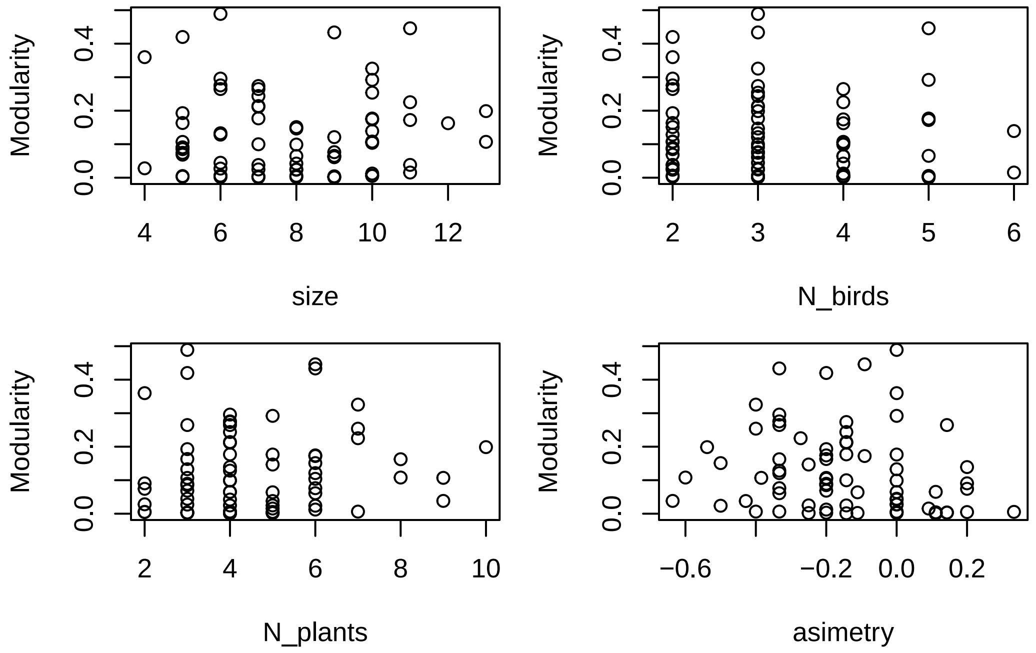

Figure S8: Scatter plots showing relationships between network metrics (y-axis), and their size (i.e., total species, N birds, N plants, asymetry)

```r
par(mfrow = c(1, 1))
```

```r
metrics_networks$month <- gsub('(.*)([0-9][0-9])$', '\\2',
                               metrics_networks$site)

metrics_networks$site2 <- gsub('([A-Z]*)(.*)$', '\\1',
                               metrics_networks$site)

metrics_networks$month <- as.numeric(metrics_networks$month)
```

S5 Interaction functional space

S5.1 Data wrangling

After verifying the reliability of the posterior distributions (Appendix S3), we compared whether observed values that were not used to fit the model were contained within the 99.5% CI of the posterior predictive simulations of each trait per species. Thus, we demonstrated that our models capture the range of variation in functional traits for birds and fruits per species (Appendix S3: sections S3.2.2.5, S3.2.3.5 and S3.2.4.5.). Then, assuming a community of 2000 individuals, we simulated the functional traits of the species considering their relative number of interactions at each network per site i and month j (Appendix S4: section 5). Finally, we used a one-class support vector machine algorithm (with a default = 0.5) (Blonder et al., 2014, 2018) to estimate the four types of IFS: bird morphology, fruit morphology, fruit nutritional content, and total fruit (nutritional plus morphological). Thus, we calculate IFS per site i during the month j (Appendix S4: section S5).

The following code uses the posterior predictive simulations of the traits per species to estimate community's interaction functional space based on the relative contribution of the species to the interactions. The code includes a function to conduct imputation of species with `NA` values.

```r
names(bird_traits) <- toupper(names(bird_traits))

bird_traits <-
  lapply(bird_traits, FUN =
           function(x) {
             x$sp <- toupper(x$sp)
             x
           })

plant_traits <-
  lapply(plant_traits, FUN =
         function(x) {
           sp <- unique(x$sp)
           indx <- which(plants_codes_full$species == sp)
           cod <- unique(plants_codes_full$code[indx])

           if (length(cod) == 1) {
             x$code <- cod
           } else {
             x$code <- cod[2]
           }
           x
         })

names(plant_traits) <-
  unlist(lapply(plant_traits, FUN = function(x) unique(x$code)))

(indx <- do.call('rbind', df)$plant %in% sort(names(plant_traits)))
```

```
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [13]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [97]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
[157]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[169]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[181]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[193]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[205]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[217]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[229]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[241]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[253]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[265]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[277]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[289]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[301]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[313]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[325]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[337]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[349]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[361]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[373]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[385]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[397]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[409]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[421]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[433]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[445]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[457]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[469]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[481]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[493]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[505]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[517]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[529]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[541]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[553]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[565]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[577]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[589]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[601]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[613]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[625]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[637]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[649]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[661]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
 [673]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [685]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [697]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [709]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [721]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [733]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [745]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [757]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [769]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [781]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [793]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
 [805] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [817]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [829]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [841]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [853]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [865]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [877]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
 [889]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [901]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [913]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [925]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [937]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [949]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [961]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [973]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [985]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [997]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1009]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1021]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1033]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1045]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1057]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1069]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1081]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1093]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1105]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1117]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1129]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[1141]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1153]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1165]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1177]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
[1189]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1201]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[1213]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[1225]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1237]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1249]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1261]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1273]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1285]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1297]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1309]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1321]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1333]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1345]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1357]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1369]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1381]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1393]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1405]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1417]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
[1429] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1441]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1453]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1465]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1477]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1489]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1501]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1513]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1525]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1537]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[1549]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1561]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1573]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1585]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1597]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1609]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1621]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1633]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1645]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1657]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1669]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1681]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[1693]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
[1705]  TRUE
```

```
plants_imputation <-
  do.call('rbind', df)$plant[!indx] |> unique() # those are the plants to
                                                # conduct imputation
df_relative_abu <-
  lapply(df, FUN =
         function(x) {

           # Here we calculated the relative abundance of plant and bird
           # species in the network. Then, we use it to calculate the
           # number of individuals per species in a simulated community
           # of 2e3 individuals

           x$year <- year(x$date)

           bird <-
             x |>
             dplyr::select(site, year, month, bird) |>
             group_by(bird, year) |>
             mutate(bird_abu = length(month)) |>
             ungroup() |> unique() |>
             mutate(bird_abu = (bird_abu/sum(bird_abu))*100,
                    ind_perc = round((2e3 * bird_abu)/100),
                    t = sum(ind_perc))

           plant <-
             x |>
             dplyr::select(site, year, month, plant) |>
             group_by(plant, year) |>
             mutate(plant_abu = length(month)) |>
             ungroup() |> unique() |>
             mutate(plant_abu = (plant_abu/sum(plant_abu))*100,
                    ind_perc = round((2e3 * plant_abu)/100),
                    t = sum(ind_perc))

           list(bird = bird,
                plant = plant)
         })

df_relative_abu <-
  lapply(df_relative_abu, FUN =
         function(x) {
           j <- lapply(x, FUN =
                         function(i) {

                           # this is necessary to guarantee that the summation
                           # of individuals in plants and birds is equal to 2e3.
                           # We need this to bind the two matrices of traits

                           tot <- unique(i$t)

                           if (tot < 2e3) {
                             i <- i[order(i$ind_perc, decreasing = T), ]
                             diff <- 2e3 - i$t[1]
                             i$ind_perc[1] <- i$ind_perc[1] + diff
                           }

                           if (tot > 2e3) {
                             i <- i[order(i$ind_perc, decreasing = T), ]
                             diff <- i$t[1] - 2e3
                             i$ind_perc[1] <- i$ind_perc[1] - diff
                           }

                           i$t2 <- sum(i$ind_perc)
                           i
```

```r
                         })
           j
         })


fun_imputation <-
  function(sp_code = plants_imputation[1], sim_ind = 1e3) {
    d <- plants_codes_full
    d2 <- d[d$code != sp_code, ]
    tax <- d[which(d$code == sp_code), ]

    # Here we detect the species matching genus, family or order
    # of the species to conduct imputation, and extract indexes to
    # index the list of simulated traits
    df_gen <- na.omit(d2[d2$genus == tax$genus, ])
    indx_gen_trait <- names(plant_traits) %in% df_gen$code
    df_fam <- na.omit(d2[d2$family == tax$family, ])
    indx_fam_trait <- names(plant_traits) %in% df_fam$code
    df_ord <- na.omit(d2[d2$order == tax$order, ])
    indx_ord_trait <- names(plant_traits) %in% df_ord$code

    traits_imputation <- # function to conduct trait imputation
      function(df) { # list of species and posterior predictions of the traits
        imputed_traits <-
          lapply(seq_along(df[[1]]), FUN =
                   function(j) {
                     trait <- do.call('cbind', # binding columns of same traits
                                      lapply(df, function(i) i[, j]))

                     trait <- apply(trait, 1, mean) # imputation trait i
                     matrix(trait[1:sim_ind], # simulated individuals
                            ncol = 1)
                   })

        # binding imputed traits on a single matrix
        imputed_traits <- do.call('cbind', imputed_traits)
        colnames(imputed_traits) <- colnames(df[[1]])
        imputed_traits
      } # end of function `traits_imputation`

    # The following statements control the imputation depending on the
    # available species at genus, family or oderd level.
    if (nrow(df_gen) > 0 & sum(indx_gen_trait) > 0) {
      message(paste0('Imputing across genus: ', unique(df_gen$genus)))
      message('...')

      d_trait <- lapply(plant_traits[indx_gen_trait], # selecting species of same genus
                        function(x) x[, 2:7])

      imputation_out <- traits_imputation(d_trait)
      imputation_out

    } else if (nrow(df_fam) > 0 & sum(indx_fam_trait) > 0) {
      message(paste0('Imputing across family: ', unique(df_fam$family)))
      message('...')

      d_trait <- lapply(plant_traits[indx_fam_trait], # selecting species of same family
                        function(x) x[, 2:7])

      imputation_out <- traits_imputation(d_trait)
      imputation_out

    } else if (nrow(df_ord) > 0 & sum(indx_ord_trait) > 0) {
      message(paste0('Imputing across order: ', unique(df_ord$order)))

      d_trait <- lapply(plant_traits[indx_ord_trait], # selecting species of same order
                        function(x) x[, 2:7])
      imputation_out <- traits_imputation(d_trait)
      imputation_out

    } else {
      message(paste0('There is not genus, family or orders to impute the species ',
                     sp_code, '. Imputing across all species.'))
      message('...')

      d_trait <- lapply(plant_traits, function(x) x[, 2:7]) # using all species

      imputation_out <- traits_imputation(d_trait)
      imputation_out

    }
  }

extract_functional_space <- # function to estimate the hypervolume
  function(month_site = 'EKA.01') {

    # first, we separete data of birds and plants
    plants <- df_relative_abu[[month_site]]$plant
    birds <- df_relative_abu[[month_site]]$bird

    # This is to detect if we have to impute one or more plant
    no_imputation <- sum(plants$plant %in% plants_imputation) == 0

    # the estimations will consider the structure of the community
    HV_birds <-
      lapply(seq_along(birds$bird), FUN =
               function(i) {
                 # extracting the traits of N individuals of
                 # the species i depending on its relative abundance

                 perc <- birds$ind_perc[i]

                 sp <- birds$bird[i]

                 indx <- which(names(bird_traits) == sp)

                 traits <- bird_traits[[indx]]

                 as.matrix(traits[1:perc, 2:4])

               })

    HV_birds <- do.call('rbind', HV_birds)

    indx_imputation <- plants$plant %in% plants_imputation

    message(paste0('We have ', sum(indx_imputation), ' sp/spp to impute'))

    if (no_imputation) {
      HV_plants <-
        lapply(seq_along(plants$plant), FUN =
                 function(i) {

                   # number of individuals to simulate
                   perc <- plants$ind_perc[i]

                   # code of the plant
                   sp <- plants$plant[i]

                   # select the spp in the posterior predictive data
                   indx <- which(names(plant_traits) == sp)

                   traits <- plant_traits[[indx]]
```

```
                # bind the simulated individuals in a
                as.matrix(traits[1:perc, 2:7])

              })

    HV_plants <- do.call('rbind', HV_plants)
  } else {
    plants_imp <- plants[indx_imputation, ] # select the species to impute

    plants_imp <- lapply(1:nrow(plants_imp), FUN =
                           function(i) {

                             # number of individuals to simulate
                             N_inds <- plants_imp$ind_perc[i]
                             # code of the plant
                             code_plant <- plants_imp$plant[i]

                             # imputation
                             fun_imputation(code_plant, sim_ind = N_inds)
                           })

    plants_imp <- do.call('rbind', plants_imp)

    plants_no_imp <- plants[!indx_imputation, ] # select spp that do not need imputation

    plants_no_imp <-
      lapply(seq_along(plants_no_imp$plant), FUN =
               function(i) {

                 # number of individuals to simulate
                 perc <- plants_no_imp$ind_perc[i]

                 # code of the plant
                 sp <- plants_no_imp$plant[i]

                 # select the spp in the posterior predictive data
                 indx <- which(names(plant_traits) == sp)

                 traits <- plant_traits[[indx]]

                 # bind the simulated individuals in a
                 as.matrix(traits[1:perc, 2:7])

               })

    plants_no_imp <- do.call('rbind', plants_no_imp)

    # binding plants with and without imputation
    HV_plants <- rbind(plants_no_imp, plants_imp)

  }

  # We bind the birds and plants matrices to generate the trait matrix
  # for the network.
  HV_network <- cbind(HV_birds, HV_plants)

  # We reduce the dimensionality of the functional space of the network to
  # improve hypervolume calculation (only gape, fruit diameter and nutritional
  # traits are considered)
  HV_network <- HV_network[, -c(1:2, 5:6)]

  # Here we index plants matrix to estimate hypervolumes of the
  # morphological and nutritional spaces
  HV_plants_morfo <- HV_plants[, 1:3]
  HV_plants_nut <- HV_plants[, 4:6]
```

```
    # Hypervolume estimations
    message('Calculating network hypervolume')
    set.seed(5)
    HV_networks <- hypervolume(HV_network, method = 'svm', svm.gamma = 0.5)

    message('Calculating plants hypervolume (total)')
    set.seed(5)
    HV_plants <- hypervolume(HV_plants, method = 'svm', svm.gamma = 0.5)

    message('Calculating plants hypervolume (morphological)')
    set.seed(5)
    HV_plants_morfo <- hypervolume(HV_plants_morfo, method = 'svm', svm.gamma = 0.5)

    message('Calculating plants hypervolume (nutritional)')
    set.seed(5)
    HV_plants_nut <- hypervolume(HV_plants_nut, method = 'svm', svm.gamma = 0.5)

    message('Calculating birds hypervolume')
    set.seed(5)
    HV_birds <- hypervolume(HV_birds, method = 'svm', svm.gamma = 0.5)

    message('Hypervolumes calculation done')
    # extract volumes
    tibble(site = month_site,
           HV_network = HV_networks@Volume,
           HV_plant = HV_plants@Volume,
           HV_plants_morfo = HV_plants_morfo@Volume,
           HV_plants_nut = HV_plants_nut@Volume,
           HV_bird = HV_birds@Volume)
  }
```

S5.2 Estimating functional space

```
t1 <- Sys.time()
hypervolumes <-
  lapply(seq_along(df_relative_abu), FUN =
             function(x) {
               i <- names(df_relative_abu)[x]
               message(paste('Running site', x,
                             'from', length(names(df_relative_abu)),
                             'sites'))
               extract_functional_space(i)
             })
Sys.time() - t1
```

```
hypervolumes <- do.call('rbind', hypervolumes)

hypervolumes$site <- names(df_relative_abu)
```

S6 Final data

```
rainfall <- readRDS('AVG_climate_data.rds')[[1]]
temperature <- readRDS('AVG_climate_data.rds')[[2]]
```

```r
sites <-
  paste(gsub('^([A-Z]*)(\\.)([0-9]*)$', '\\1', hypervolumes$site),
      as.numeric(gsub('^([A-Z]*)(\\.)([0-9]*)$', '\\3', hypervolumes$site)),
      sep = '_')

hypervolumes$site <- sites

metrics_networks$site <- metrics_networks %$% paste(site2, month, sep = '_')

colnames(rainfall)[1] <- 'z_rainfall'

rainfall$z_rainfall <- as.vector(scale(rainfall$z_rainfall))

rainfall$site <-
  rainfall %$% paste(sites, month, sep = '_')

colnames(temperature)[1] <- 'z_temperature'
temperature$z_temperature <- as.vector(scale(temperature$z_temperature))

temperature$site <-
  temperature %$% paste(sites, month, sep = '_')

dat <- full_join(metrics_networks, hypervolumes, by = 'site')

dat <- left_join(dat,
                 temperature[, c("site", "z_temperature")],
                 by = 'site')

left_join(dat,
          rainfall[, c("site", "z_rainfall")],
          by = 'site') |>
  apply(2, function(x) sum(is.na(x)))
```

```
         size         N_birds        N_plants        asimetry      nestedness
            0               0               0               0               0
   modularity              H2            site           month           site2
            0               0               0               0               0
   HV_network        HV_plant HV_plants_morfo    HV_plants_nut         HV_bird
            0               0               0               0               0
z_temperature      z_rainfall
            0               0
```

```r
dat <-
  left_join(dat,
           rainfall[, c("site", "z_rainfall")],
           by = 'site')

#saveRDS(dat, 'data_for_models.rds')
```

S7 Computational environment

```r
sessionInfo()
```

```
R version 4.5.1 (2025-06-13)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.5

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Sao_Paulo
tzcode source: internal

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] readxl_1.4.5         hypervolume_3.1.6     Rcpp_1.1.0
 [4] iNEXT_3.0.1          igraph_2.1.4          bipartite_2.21
 [7] vegan_2.7-1          permute_0.9-8         sna_2.8
[10] network_1.19.0       statnet.common_4.12.0 cowplot_1.2.0
[13] rethinking_2.42      posterior_1.6.1       cmdstanr_0.9.0.9000
[16] magrittr_2.0.3       forcats_1.0.0         lubridate_1.9.4
[19] ggplot2_3.5.2        dplyr_1.1.4

loaded via a namespace (and not attached):
  [1] pROC_1.18.5         rlang_1.1.6         e1071_1.7-16
  [4] matrixStats_1.5.0   compiler_4.5.1      mgcv_1.9-3
  [7] loo_2.8.0           vctrs_0.6.5         maps_3.4.3
 [10] reshape2_1.4.4      stringr_1.5.1       pkgconfig_2.0.3
 [13] shape_1.4.6.1       crayon_1.5.3        fastmap_1.2.0
 [16] backports_1.5.0     labeling_0.4.3      magic_1.6-1
 [19] rmarkdown_2.29      prodlim_2025.04.28  pracma_2.4.4
 [22] ps_1.9.1            purrr_1.0.4         xfun_0.52
 [25] jsonlite_2.0.0      progress_1.2.3      recipes_1.3.1
 [28] pdist_1.2.1         terra_1.8-54        prettyunits_1.2.0
 [31] cluster_2.1.8.1     R6_2.6.1            stringi_1.8.7
 [34] RColorBrewer_1.1-3  extrafontdb_1.0     parallelly_1.45.0
 [37] rpart_4.1.24        cellranger_1.1.0    iterators_1.0.14
 [40] knitr_1.50          future.apply_1.20.0 fields_16.3.1
 [43] extrafont_0.19      Matrix_1.7-3        splines_4.5.1
 [46] nnet_7.3-20          timechange_0.3.0     tidyselect_1.2.1
 [49] rstudioapi_0.17.1    abind_1.4-8          yaml_2.3.10
 [52] timeDate_4041.110    doParallel_1.0.17    codetools_0.2-20
 [55] processx_3.8.6       listenv_0.9.1        lattice_0.22-7
 [58] tibble_3.3.0         plyr_1.8.9           ks_1.15.1
 [61] withr_3.0.2          coda_0.19-4.1        evaluate_1.0.4
 [64] future_1.58.0        survival_3.8-3       proxy_0.4-27
 [67] mclust_6.1.1         pillar_1.11.0        tensorA_0.36.2.1
 [70] KernSmooth_2.23-26   stats4_4.5.1         checkmate_2.3.2
 [73] foreach_1.5.2        geometry_0.5.2       distributional_0.5.0
 [76] generics_0.1.4       hms_1.1.3            scales_1.4.0
 [79] globals_0.18.0       class_7.3-23         glue_1.8.0
 [82] tools_4.5.1          data.table_1.17.8    ModelMetrics_1.2.2.2
 [85] gower_1.0.2          mvtnorm_1.3-3        fastcluster_1.3.0
 [88] dotCall64_1.2        grid_4.5.1           Rttf2pt1_1.3.12
 [91] ipred_0.9-15         nlme_3.1-168         palmerpenguins_0.1.1
 [94] cli_3.6.5            spam_2.11-1          viridisLite_0.4.2
 [97] lava_1.8.1           gtable_0.3.6         digest_0.6.37
[100] caret_7.0-1          farver_2.1.2         htmltools_0.5.8.1
[103] lifecycle_1.0.4      hardhat_1.4.1        MASS_7.3-65
```