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

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

Appendix S1. Journal: Ecology

Table of contents

S1 Introduction

This document reproduces the Bayesian models estimating functional traits of birds and plants from **Ramírez-Mejía et al. (2023) 'Temporal dynamics of seed-dispersal networks: Disentangling the role of direct and indirect biotic and climatic effects'**. It is organized into two main sections: (1) plant traits and (2) bird traits. Each section includes:

- Data collection
- The mathematical notation of the model
- Stan code for model fitting
- Fitting procedures
- Sampling diagnostics
- Visualizations of posterior distributions.

S2 Data collection

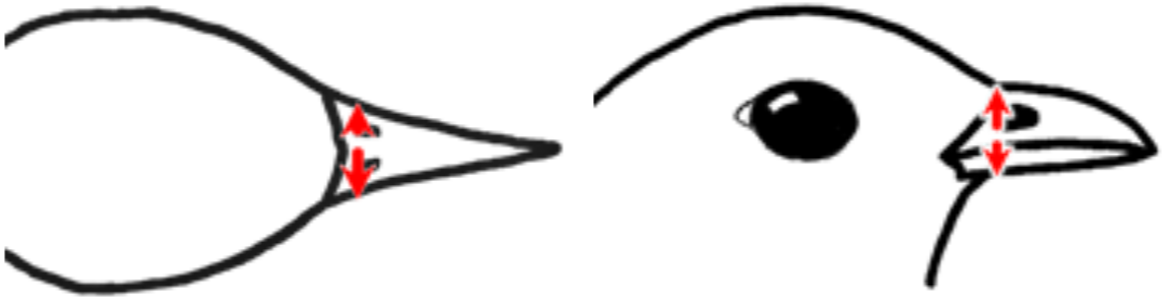
S2.1 Bird traits

Mist netting was conducted at each site from November 2014 through December 2017, to record bill gape (mm) and bill depth (mm) as show in figures below. We also recorded mass of each individual by weighing the birds using spring scales (PESOLA, Switzerland) in a bag and subtracting the mass of the bag. Morphological measurements of juvenile birds were not used in any analysis.

We measured three morphological traits of frugivorous birds: body mass, bill depth, and bill gape. Body mass is related to the species' vagility and frugivory rate (Case & Tarwater 2020; Wotton & Kelly 2012), whereas bill depth and gape impose mechanical constraints on the fruit choices (but see (Rojas et al. 2025))

S2.2 Plant

Fruit collections typically involved multiple fruits from several individuals (mean ± SD, 18.17 ± 1.68 fruits per species). Fruit and seed measurements for some rare species were based on collections from ex-situ living plant collections at the US Army Natural Resources Program



(a) Bill gape (b) Bill depth

Figure S1: Schemes showing measurements of bill traits recorded in O'ahu island (schemes were taken from supplementary material in Gleditsch, J. M., & Sperry, J. H. (2019). Rapid morphological change of nonnative frugivores on the Hawaiian island of O'ahu*. Evolution, 73(7), 1456–1465. https://doi.org/10.1111/evo.13744)

on O'ahu (ANRP-O) or other local botanical sources. Occasionally, measurements were made from stored seed collections at ANRP-O's Seed Conservation Laboratory and seed and fruit data were queried from the ANRP-O database. Because only seeds from mature fruits were included in measurements, we assume that the natural drying of seeds throughout maturation would allow unbiased comparisons between fresh and dried/stored seeds.

S2.2.1 Morphological traits

Fresh fruits were measured with calipers to obtain morphological data. Length was designated as the longest axis and width the shortest axis. We collected data on seed morphology (length and width) using calipers or an ocular micrometer on a dissecting microscope. Seeds were measured along two axes, with the longest being designated as length and the next longest as width. In most cases, the number of seeds/fruit was averaged from 10 whole, ripe fruits, although the number of fruits measured per species 9.61 $\pm$ 0.75.

S2.2.2 Nutritional traits

For nutritional analyses, fruits were dried in a commercial food dehydrator at a temperature no greater than 48°C and weighed daily until mass remained constant. After all seeds were removed, fruits were again dried to constant mass, then ground to a fine powder in a Cyclone Lab Sample Mill (Udy Corporation). The number of fruits examined per species for nutritional analyses, depended on size of fruit and rarity, with an average of 105.39 $\pm$ 24.48 fruits per species. The following quantities of fruit material were typically used for analyses: approximately 500 mg for lipids, 100 mg for watersoluble carbohydrates (WSC), and 250 mg for

protein. However, for species having limited material, smaller quantities were analyzed. The minimum quantities used were 207 mg for lipids, 81 mg for WSC, and 41 mg for protein.

WSC content was determined colorimetrically using the anthrone method. WSC were extracted by boiling fruit material in deionized water (10 mL water per 100 mg fruit) for six minutes and the resulting solution was diluted to 100 mL with deionized water. Then, two 0.1 mL aliquots were taken, diluted to 1 mL with deionized water, and added to a 2 mg/mL solution of anthrone in 95% sulfuric acid (total of 5 mL). Test tubes containing this solution were immersed in boiling water for eight minutes and the resulting reaction produced a green color. Absorbance of each aliquot at 630 nm was read with a spectrophotometer, and the mean absorbance was used to determine WSC content via comparison to a glucose standard curve.

Lipid content was determined using the Folch method. Fruit material was combined with a 2:1 chloroform–methanol solution (2 mL solution per 100 mg fruit) and agitated on a rocker for 20 min, followed by centrifugation. The solution was then removed from the solid fruit material, washed by vortexing with a 0.9% NaCl solution (2 mL), and the final chloroform/methanol solution evaporated on a pre-weighed aluminum dish to leave the lipid residue, which was then weighed to obtain mass.

Nitrogen content was determined by combustion using AOCS Method Ba 4e-93 with a Flash 1112 Protein Analyzer (CE Elantech) at the laboratory of the Illinois Crop Improvement Association in Champaign, IL. **Crude protein content** was estimated by multiplying nitrogen content by 5.64

Those trait are relevant since birds may condition their fruit choice depending on the size of the fruit that they can process and manipulate (Rojas et al. 2025), while seed number can affect the amount of ingested fruits depending on the digestive capability of the bird (Godínez-Alvarez et al. 2020). Moreover, fruit-eating birds can modulate their foraging preferences depending on their nutritional requirement, either seeking a balanced diet (Blendinger et al. 2022), or by favoring a particular macronutrient (Morán-López et al. 2025).

NOTE: For further details see Sperry, J. H., O'Hearn, D., Drake, D. R., Hruska, A. M., Case, S. B., Vizentin-Bugoni, J., Arnett, C., Chambers, T., & Tarwater, C. E. (2021). Fruit and seed traits of native and invasive plant species in Hawai'i: implications for seed dispersal by non-native birds. Biological Invasions, 23(6), 1819–1835. https://doi.org/10.1007/s10530-021-02473-z

S3 Data analysis

Using the fruit and bird traits, we calculated four types of IFS: bird morphological, fruit morphological, fruit nutritional, and fruit total, which is a combination of nutritional and morphological traits. Estimating the IFS is data-demanding (Blonder, 2018); therefore, we modeled each trait using hierarchical models and used posterior predictive simulations to

generate random values for species traits. First, we averaged trait values per species to estimate their parameters using the same amount of data. Then, we modeled z-scores of each trait, including the species as population effect and the genus, family, and order as grouping effects to account for phylogenetic relatedness among species (Supporting Information S3: sections 3). In the case of plant models, the species status (i.e., native, alien, or invasive) was also included as a grouping factor. We used a normal or Student's t-distribution as likelihood functions depending on the trait, and employed partial pooling to estimate the parameters of group-level effects. We estimated the posterior distribution of the parameters and conducted sampling diagnostics as described above (see Supporting Information S3: sections S3.1–S3.2 for a detailed mathematical version of the models, prior probabilities employed, Stan code, and sampling diagnostic procedures).

After verifying the reliability of the posterior distributions, 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 (Supporting Information 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 (Supporting Information 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 (Supporting Information S4: section S5).

S3.1 Plant traits

The following code conducts data wrangling, formatting for model fitting, and identification of `NA` values for imputation.

```
sapply(c('dplyr', 'forcats', 'lubridate',
         'magrittr', 'rethinking', 'cmdstanr',
         'cowplot', 'readxl', 'ggplot2', 'sf'), library, character.only = T)

extrafont::loadfonts(device = 'win')

source('functions_mod_diagnostics.r')

plant_traits <- read_xlsx('plant_traits/PlantTraits_v1.xlsx',
                          sheet = 1, col_names = T)

str(plant_traits)

nom <- colnames(plant_traits)

plant_traits <-
  do.call('cbind',
          lapply(plant_traits, FUN =
                   function(x) {
                     i <- sum(grepl('[0-9]', x))
                     if (i > 0) tibble(as.numeric(x))
```

```
                    else tibble(x)
                  }))

colnames(plant_traits) <- nom

# saveRDS(plant_traits[, c("code", "species", "family",
#                          "order", "origin", "status")],
#         'plants_codes.rds')

plant_traits <-
  plant_traits[, c("species", "family", "order", "origin", "status",
                   "fwidthmean", "swidthmean", "seedsfruit", "wscarbs",
                   "lipid", "protein")]

for (i in 1:5) {
  plant_traits[[i]] <- as.factor(plant_traits[[i]])
}

dat_plant <-
  lapply(plant_traits, FUN =
           function(i) {
             if (is.factor(i)) as.numeric(i)
             else i
           })

dat_plant$N <- length(dat_plant$species)
dat_plant$N_spp <- max(dat_plant$species)
dat_plant$N_family <- max(dat_plant$family)
dat_plant$N_order <- max(dat_plant$order)
dat_plant$N_origin <- max(dat_plant$origin)
dat_plant$N_status <- max(dat_plant$status)
dat_plant$na_f_width <- which(is.na(dat_plant$fwidthmean))
dat_plant$N_na_f_width <- length(dat_plant$na_f_width)
dat_plant$na_s_width <- which(is.na(dat_plant$swidthmean))
dat_plant$N_na_s_width <- length(dat_plant$na_s_width)
dat_plant$na_s_number <- which(is.na(dat_plant$seedsfruit))
dat_plant$N_na_s_number <- length(dat_plant$na_s_number)
dat_plant$na_carbs <- which(is.na(dat_plant$wscarbs))
dat_plant$N_na_carbs <- length(dat_plant$na_carbs)
dat_plant$na_lipid <- which(is.na(dat_plant$lipid))
dat_plant$N_na_lipid <- length(dat_plant$na_lipid)
dat_plant$na_protein <- which(is.na(dat_plant$protein))
dat_plant$N_na_protein <- length(dat_plant$na_protein)

for (i in 6:11) {
  dat_plant[[i]][which(!is.na(dat_plant[[i]]))] <-
    as.vector(log(dat_plant[[i]][which(!is.na(dat_plant[[i]]))]))
  dat_plant[[i]][which(is.na(dat_plant[[i]]))] <- 0
}


dat_plant$fwidthmean[which(!is.na(plant_traits$fwidthmean))] <-
   as.vector(scale(dat_plant$fwidthmean[which(!is.na(plant_traits$fwidthmean))]))
```

S3.1.1 Mathematical notation of the models

These models estimate species-specific variation in *fruit diameter*, *seed diameter*, and *seeds per fruit* among fruiting plants, while accounting for taxonomic effects (family and order), origin, invasive status, and imputation of `NA` values in the response variable. All models share the same structure; thus, the following mathematical notation applies to each.

Let $\mathbf{y_i^{obs}}$ be observed values of the response variables, and $\mathbf{y_i^{miss}}$ missing values at the indices $\mathcal{M}$. The merged vector $\mathbf{y_i^{merged}}$ is defined as:

$$y_i^{\text{merged}} = \begin{cases} y_i^{\text{obs}} & \text{if } i \notin \mathcal{M}_i \\ y_i^{\text{mis}} & \text{if } i = \mathcal{M}_i \end{cases}$$

Imputation:

$$\begin{aligned} &\mu_{\text{imputation}} \sim \mathcal{N}(0, 0.5) \\ &\sigma_{\text{imputation}} \sim Exp(1) \\ &y_i^{mis} \sim \mathcal{N}(\mu_{\text{imputation}}, \sigma_{\text{imputation}}) \end{aligned}$$

Full model

$$\begin{aligned} &y_i^{\text{merged}} \sim \textit{Student-T}(\nu = 7, \mu_i, \sigma) \\ &\mu_i = \alpha_{sp} + \theta_{family} + \phi_{order} + \delta_{origin} + \gamma_{status} \\ &\alpha_{sp} \sim \mathcal{N}(0, 1) \\ &\theta_{family} = \mu_\theta + Z_\theta \cdot \sigma_\theta \\ &\phi_{order} = \mu_\phi + Z_\phi \cdot \sigma_\phi \\ &\delta_{origin} = \mu_\delta + Z_\delta \cdot \sigma_\delta \\ &\gamma_{status} = \mu_\gamma + Z_\gamma \cdot \sigma_\gamma \\ &\mu_{\theta,\phi,\delta,\gamma} \sim \mathcal{N}(0, 1) \\ &Z_{\theta,\phi,\delta,\gamma} \sim \mathcal{N}(0, 1) \\ &\sigma_{\theta,\phi,\delta,\gamma} \sim Exp(0, 1) \\ &\sigma \sim Exp(1) \end{aligned}$$

S3.1.2 Fruit diameter

S3.1.2.1 Stan code

```
cat(file = 'fruit_diameter.stan',
    '
    functions{
      vector merge_missing(array[] int miss_indxex, vector x_obs, vector x_miss) {
              int N = dims(x_obs)[1];
              int N_miss = dims(x_miss)[1];
              vector[N] merge;
              merge = x_obs;
              for (i in 1:N_miss) {
                  merge[miss_indxex[i]] = x_miss[i];
```

```
          }
      return merge;
  }
}

data{
  int N;
  int N_spp;
  int N_family;
  int N_order;
  int N_origin;
  int N_status;
  int N_na_f_width;
  int N_na_s_width;
  int N_na_s_number;
  int N_na_carbs;
  int N_na_lipid;
  int N_na_protein;
  array[N_na_f_width] int na_f_width;
  array[N_na_s_width] int na_s_width;
  array[N_na_s_number] int na_s_number;
  array[N_na_carbs] int na_carbs;
  array[N_na_lipid] int na_lipid;
  array[N_na_protein] int na_protein;
  array[N] int species;
  array[N] int family;
  array[N] int order;
  array[N] int origin;
  array[N] int status;
  vector[N] fwidthmean;
}

parameters{

  // imputed fwidthmean
  vector[N_na_f_width] y_imputed;
  real mu_imputed;
  real<lower = 0> sigma_imputed;

  vector[N_spp] spp;
  //real mu_spp;
  //real<lower = 0> sigma_spp;

  vector[N_family] z_fam;
  real mu_fam;
  real<lower = 0> sigma_fam;

  vector[N_order] z_ord;
  real mu_ord;
  real<lower = 0> sigma_ord;

  vector[N_origin] z_ori;
  real mu_ori;
  real<lower = 0> sigma_ori;

  vector[N_status] z_sta;
  real mu_sta;
  real<lower = 0> sigma_sta;

  real<lower = 0> sigma;

}

transformed parameters{
  vector[N] y_merged;
  y_merged = merge_missing(na_f_width,
                           to_vector(fwidthmean),
```

```
                              y_imputed);

  //vector[N_spp] spp;
  //spp = mu_spp + z_spp * sigma_spp;

  vector[N_family] fam;
  fam = mu_fam + z_fam * sigma_fam;

  vector[N_order] ord;
  ord = mu_ord + z_ord * sigma_ord;

  vector[N_origin] ori;
  ori = mu_ori + z_ori * sigma_ori;

  vector[N_status] sta;
  sta = mu_sta + z_sta * sigma_sta;
}

model{
  mu_imputed ~ normal(0, 0.5);
  sigma_imputed ~ exponential(1);
  y_imputed ~ normal(0, 0.5);
  y_merged ~ normal(mu_imputed, sigma_imputed);

  //z_spp ~ normal(0, 1);
  spp ~ normal(0, 1);
  //sigma_spp ~ exponential(1);

  z_fam ~ normal(0, 1);
  mu_fam ~ normal(0, 1);
  sigma_fam ~ exponential(1);

  z_ord ~ normal(0, 1);
  mu_ord ~ normal(0, 1);
  sigma_ord ~ exponential(1);

  z_ori ~ normal(0, 1);
  mu_ori ~ normal(0, 1);
  sigma_ori ~ exponential(1);

  z_sta ~ normal(0, 1);
  mu_sta ~ normal(0, 1);
  sigma_sta ~ exponential(1);

  sigma ~ exponential(1);

  for (i in 1:N) {
    y_merged[i] ~ student_t(7, spp[species[i]] +
                        fam[family[i]] +
                        ord[order[i]] +
                        ori[origin[i]] +
                        sta[status[i]], sigma);
  }

}

generated quantities{

  array[N] real ppcheck;
  vector[N] mu;

  for (i in 1:N) {
    mu[i] = spp[species[i]] +
            fam[family[i]] +
            ord[order[i]] +
            ori[origin[i]] +
            sta[status[i]];
```

```
      }

      ppcheck = student_t_rng(7, mu, sigma);

    }

    ')
```

S3.1.2.2 Fitting the model

```
file <- paste0(getwd(), '/fruit_diameter.stan')
fit_fdiam <- cmdstan_model(file, compile = T)

mod_fdiam <-
  fit_fdiam$sample(
    data = dat_plant,
    iter_warmup = 500,
    iter_sampling = 5e3,
    thin = 3,
    chains = 3,
    parallel_chains = 3,
    seed = 555
  )
```

S3.1.2.3 Sampling diagnostics

```
sum_fdiam <- mod_fdiam$summary()

mod_diagnostics(mod_fdiam, sum_fdiam)
```

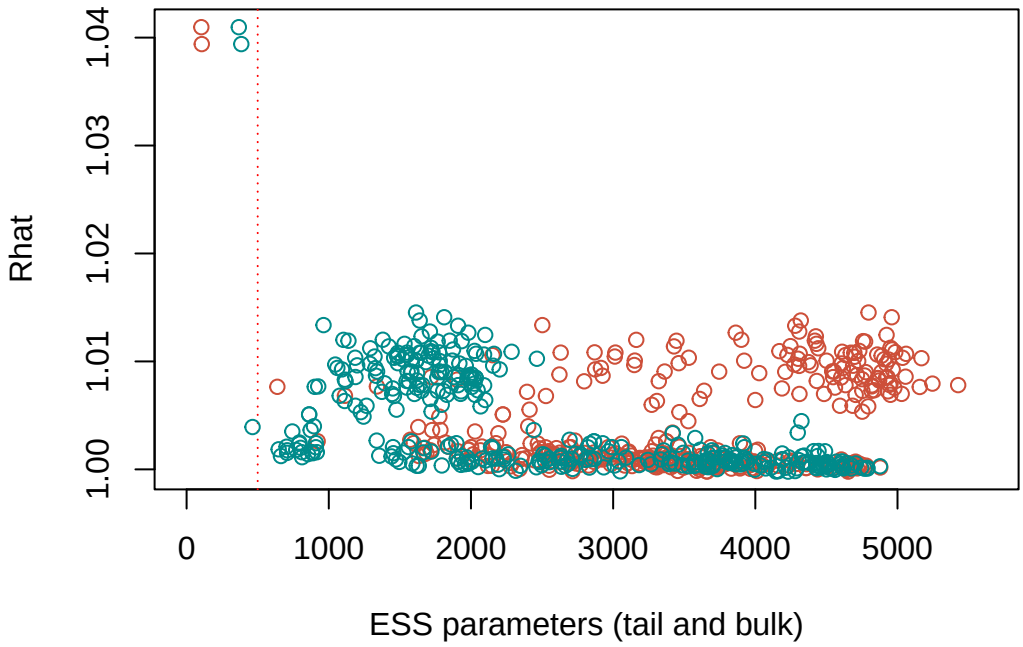


Figure S2: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_fdiam <- mod_fdiam$draws('ppcheck', format = 'matrix')

plot(density(dat_plant$fwidthmean[dat_plant$fwidthmean != 0]),
     xlab = 'Fruit diameter', ylim = c(0, 0.8), xlim = c(-8, 8),
     main = '')
for (i in 1:100) lines(density(ppcheck_fdiam[i, ]))
lines(density(dat_plant$fwidthmean[dat_plant$fwidthmean != 0]),
      col = 'red', lwd = 3)
```

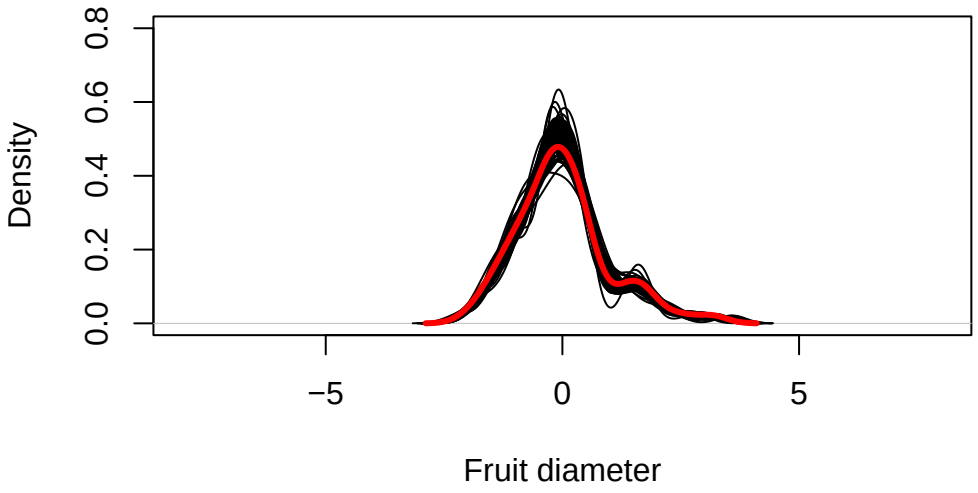


Figure S3: Posterior predictive checks of the model describing fruit diameter among species.

S3.1.2.4 Extracting posterior draws

```
# function for extracting posterior draws
extract_post_plants <-
  function(model = mod_fdiam, nu = 7, dist = 'rstudent',
           pars = c('spp', 'fam', 'ord', 'ori', 'sta', 'sigma')) {

    p <- model$draws(pars, format = 'df')

    p <-
      lapply(pars, FUN =
             function(x) {
               p[, grep(x, colnames(p))]
             })

    names(p) <- pars

    est <-
      lapply(seq_along(dat_plant$species), FUN =
               function(x) {

                 sp <- dat_plant$species[x]
                 f <- dat_plant$family[x]
                 orden <- dat_plant$order[x]
                 origin <- dat_plant$origin[x]
                 status <- dat_plant$status[x]

                 mu <- with(p,
                            {
                              spp[, sp, drop = T] +
                                fam[, f, drop = T] +
```

```
                                ord[, orden, drop = T] +
                                ori[, origin, drop = T] +
                                sta[, status, drop = T]
                            })
                  q1 <- quantile(mu, 0.0025)
                  q2 <- quantile(mu, 0.9975)
                  mu <- mu[mu >= q1 & mu <= q2]

                  if (dist == 'none') {
                    set.seed(123)
                    matrix(mu[1:2e3], ncol = 1)
                  }
                  if (dist == 'rstudent') {
                    set.seed(123)
                    matrix(rstudent(2e3, nu, mu[1:2e3], p$sigma[[1]]), ncol = 1)
                  } else {
                    set.seed(123)
                    matrix(rnorm(2e3, mu[1:2e3], p$sigma[[1]]), ncol = 1)
                  }

                })

    do.call('cbind', est)

  }

est_fdiam <- extract_post_plants(mod_fdiam, dist = 'none')
```

S3.1.2.5 Plotting the posterior distribution

```
rm('ppcheck_fdiam')

plot(NULL, xlim = c(0, 70), ylim = c(-6, 6),
     xlab = 'Plant species',
     ylab = 'Fruit diameter (z-scores)')
segments(y0 = apply(est_fdiam, 2, quantile, 0),
         y1 = apply(est_fdiam, 2, quantile, 1),
         x0 = 1:69)
points(1:69, dat_plant$fwidthmean)
points(1:69, apply(est_fdiam, 2, mean), pch = 16, col = 'red')
points(dat_plant$na_f_width,
       apply(est_fdiam[, dat_plant$na_f_width], 2, mean),
       pch = 16, col = 'cyan4')
```

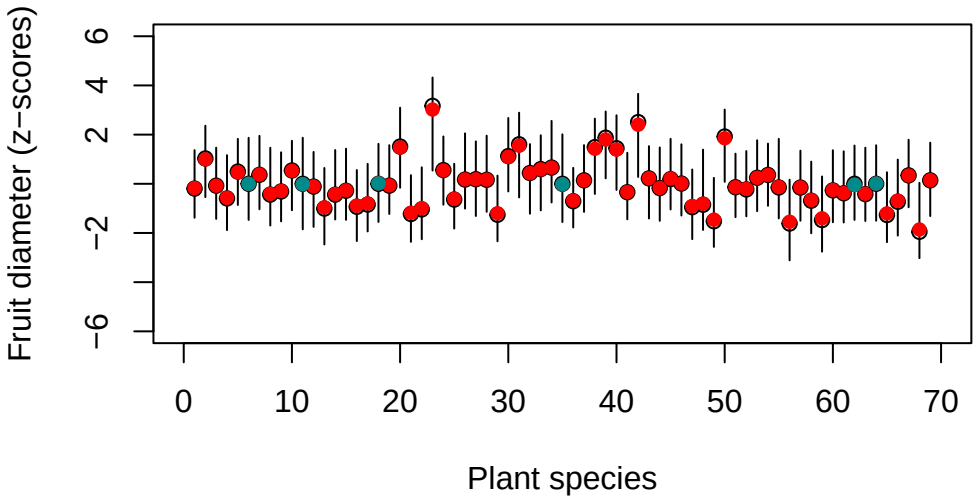


Figure S4: Posterior distributions of fruit diameter among fruiting species involve in seed dispersal networks in O'ahu island. Error bars indicate 95% CI, red dotes average values and green dotes shows imputed values

S3.1.3 Seed diameter

S3.1.3.1 Stan code

```
dat_plant$swidthmean[which(!is.na(plant_traits$swidthmean))] <-
  as.vector(scale(dat_plant$swidthmean[which(!is.na(plant_traits$swidthmean))]))
```

```
cat(file = 'seed_diameter.stan',
    '
    functions{
      vector merge_missing(array[] int miss_indxex, vector x_obs, vector x_miss) {
              int N = dims(x_obs)[1];
              int N_miss = dims(x_miss)[1];
              vector[N] merge;
              merge = x_obs;
              for (i in 1:N_miss) {
                  merge[miss_indxex[i]] = x_miss[i];
              }
          return merge;
      }
    }

    data{
      int N;
      int N_spp;
```

```
  int N_family;
  int N_order;
  int N_origin;
  int N_status;
  int N_na_f_width;
  int N_na_s_width;
  int N_na_s_number;
  int N_na_carbs;
  int N_na_lipid;
  int N_na_protein;
  array[N_na_f_width] int na_f_width;
  array[N_na_s_width] int na_s_width;
  array[N_na_s_number] int na_s_number;
  array[N_na_carbs] int na_carbs;
  array[N_na_lipid] int na_lipid;
  array[N_na_protein] int na_protein;
  array[N] int species;
  array[N] int family;
  array[N] int order;
  array[N] int origin;
  array[N] int status;
  vector[N] swidthmean;
}

parameters{

  // imputed swidthmean
  vector[N_na_s_width] y_imputed;
  real mu_imputed;
  real<lower = 0> sigma_imputed;

  vector[N_spp] spp;
  //real mu_spp;
  //real<lower = 0> sigma_spp;

  vector[N_family] z_fam;
  real mu_fam;
  real<lower = 0> sigma_fam;

  vector[N_order] z_ord;
  real mu_ord;
  real<lower = 0> sigma_ord;

  vector[N_origin] z_ori;
  real mu_ori;
  real<lower = 0> sigma_ori;

  vector[N_status] z_sta;
  real mu_sta;
  real<lower = 0> sigma_sta;

  real<lower = 0> sigma;

}

transformed parameters{
  vector[N] y_merged;
  y_merged = merge_missing(na_s_width,
                           to_vector(swidthmean),
                           y_imputed);

  //vector[N_spp] spp;
  //spp = mu_spp + z_spp * sigma_spp;

  vector[N_family] fam;
  fam = mu_fam + z_fam * sigma_fam;
```

```
  vector[N_order] ord;
  ord = mu_ord + z_ord * sigma_ord;

  vector[N_origin] ori;
  ori = mu_ori + z_ori * sigma_ori;

  vector[N_status] sta;
  sta = mu_sta + z_sta * sigma_sta;
}

model{
  mu_imputed ~ normal(0, 0.5);
  sigma_imputed ~ exponential(1);
  y_imputed ~ normal(0, 1);
  y_merged ~ normal(mu_imputed, sigma_imputed);

  spp ~ normal(0, 1);
  //mu_spp ~ normal(0, 1);
  //sigma_spp ~ exponential(1);

  z_fam ~ normal(0, 0.5);
  mu_fam ~ normal(0, 0.5);
  sigma_fam ~ exponential(1);

  z_ord ~ normal(0, 0.5);
  mu_ord ~ normal(0, 0.5);
  sigma_ord ~ exponential(1);

  z_ori ~ normal(0, 1);
  mu_ori ~ normal(0, 1);
  sigma_ori ~ exponential(1);

  z_sta ~ normal(0, 1);
  mu_sta ~ normal(0, 1);
  sigma_sta ~ exponential(1);

  sigma ~ exponential(1);

  for (i in 1:N) {
    y_merged[i] ~ student_t(7, spp[species[i]] +
                        fam[family[i]] +
                        ord[order[i]] +
                        ori[origin[i]] +
                        sta[status[i]], sigma);
  }

}

generated quantities{

  array[N] real ppcheck;
  vector[N] mu;

  for (i in 1:N) {
    mu[i] = spp[species[i]] +
            fam[family[i]] +
            ord[order[i]] +
            ori[origin[i]] +
            sta[status[i]];
  }

  ppcheck = student_t_rng(7, mu, sigma);

}

')
```

S3.1.3.2 Fitting the model

```
file <- paste0(getwd(), '/seed_diameter.stan')
fit_sdiam <- cmdstan_model(file, compile = T)

mod_sdiam <-
  fit_sdiam$sample(
    data = dat_plant,
    iter_warmup = 1e3,
    iter_sampling = 15e3,
    thin = 3,
    chains = 3,
    parallel_chains = 3,
    seed = 555
  )
```

S3.1.3.3 Sampling diagnostics

```
sum_sdiam <- mod_sdiam$summary()

mod_diagnostics(mod_sdiam, sum_sdiam)
```

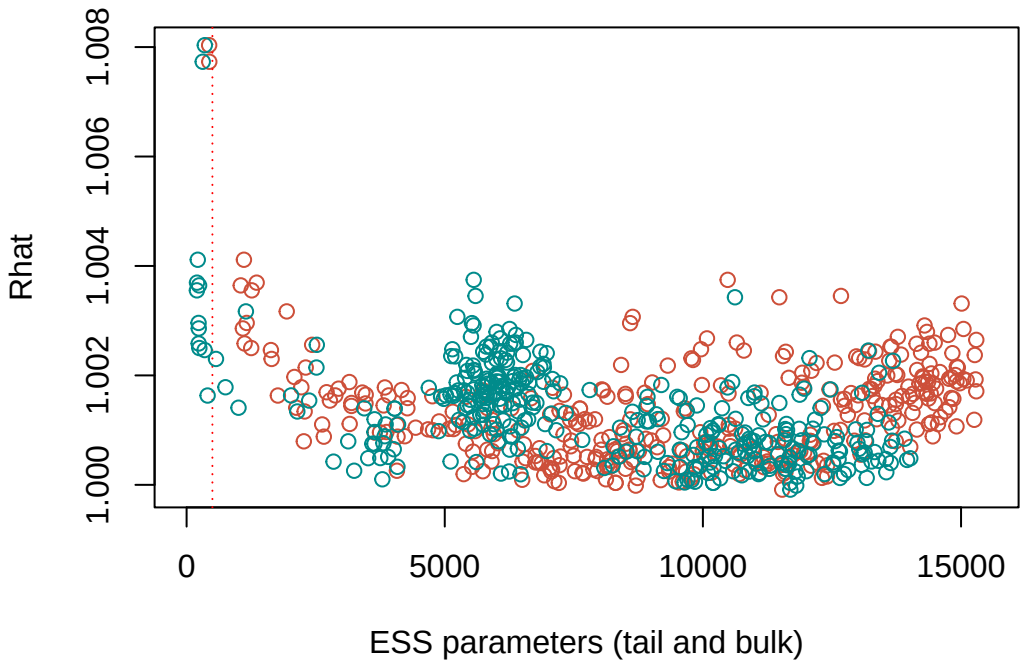


Figure S5: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_sdiam <- mod_sdiam$draws('ppcheck', format = 'matrix')

plot(density(dat_plant$swidthmean[dat_plant$swidthmean != 0]),
     xlab = 'Seed diameter',  ylim = c(0, 0.5), xlim = c(-5, 5),
     main = '')
for (i in 1:100) lines(density(ppcheck_sdiam[i, ]))
lines(density(dat_plant$swidthmean[dat_plant$swidthmean != 0]),
      col = 'red', lwd = 3)
```

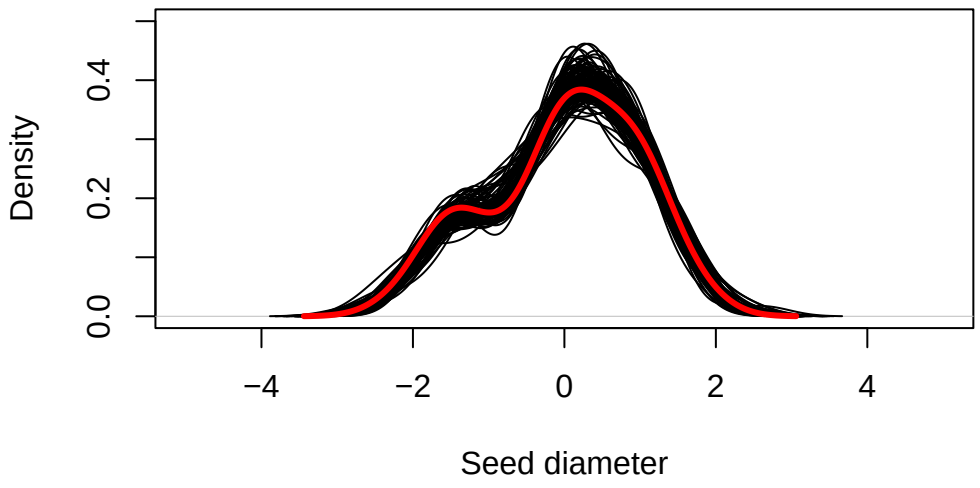


Figure S6: Posterior predictive checks of the model describing seed diameter among species.

S3.1.3.4 Extracting posterior draws

```
est_sdiam <- extract_post_plants(mod_sdiam, dist = 'none')
```

S3.1.3.5 Plotting the posterior distribution

```
rm('ppcheck_fdiam')
```

```
Warning in rm("ppcheck_fdiam"): object 'ppcheck_fdiam' not found
```

```
plot(NULL, xlim = c(0, 70), ylim = c(-5, 4),
     xlab = 'Plant species',
     ylab = 'Seeds diameter (z-scores)')
segments(y0 = apply(est_sdiam, 2, quantile, 0),
         y1 = apply(est_sdiam, 2, quantile, 1),
         x0 = 1:69)
points(1:69, dat_plant$swidthmean)
points(1:69, apply(est_sdiam, 2, mean), pch = 16, col = 'red')
points(dat_plant$na_s_width,
       apply(est_sdiam[, dat_plant$na_s_width], 2, mean),
       pch = 16, col = 'cyan4')
```

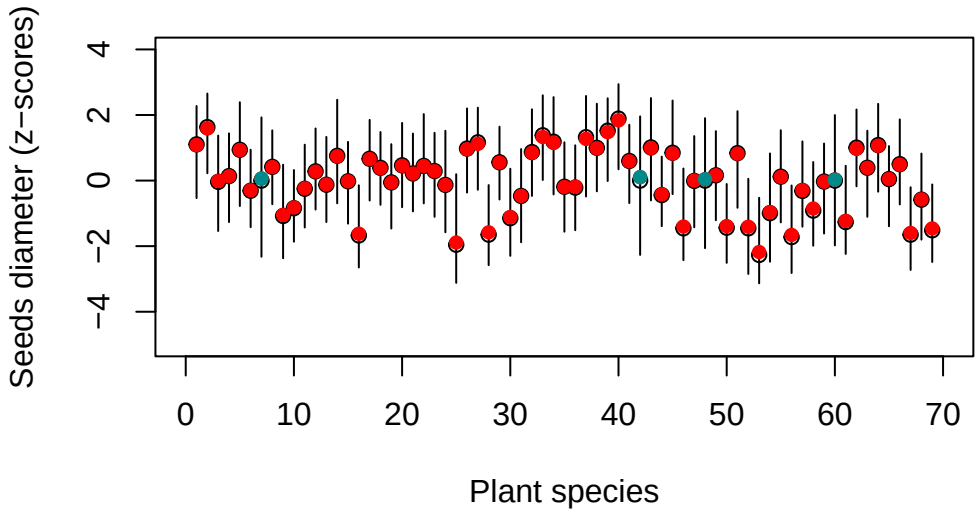


Figure S7: Posterior distributions of seeds diameter among fruiting species involve in seed dispersal networks in O'ahu island. Error bars indicate 95% CI, red dotes average values and green dotes shows imputed values

S3.1.4 Seeds per fruit

S3.1.4.1 Stan code

```
dat_plant$seedsfruit[which(!is.na(plant_traits$seedsfruit))] <-
  as.vector(scale(dat_plant$seedsfruit[which(!is.na(plant_traits$seedsfruit))]))
```

```
cat(file = 'seeds_fruit.stan',
    '
    functions{
```

```
  vector merge_missing(array[] int miss_indxex, vector x_obs, vector x_miss) {
          int N = dims(x_obs)[1];
          int N_miss = dims(x_miss)[1];
          vector[N] merge;
          merge = x_obs;
          for (i in 1:N_miss) {
              merge[miss_indxex[i]] = x_miss[i];
          }
      return merge;
  }
}

data{
  int N;
  int N_spp;
  int N_family;
  int N_order;
  int N_origin;
  int N_status;
  int N_na_f_width;
  int N_na_s_width;
  int N_na_s_number;
  int N_na_carbs;
  int N_na_lipid;
  int N_na_protein;
  array[N_na_f_width] int na_f_width;
  array[N_na_s_width] int na_s_width;
  array[N_na_s_number] int na_s_number;
  array[N_na_carbs] int na_carbs;
  array[N_na_lipid] int na_lipid;
  array[N_na_protein] int na_protein;
  array[N] int species;
  array[N] int family;
  array[N] int order;
  array[N] int origin;
  array[N] int status;
  vector[N] seedsfruit;
}

parameters{

  // imputed seedsfruit
  vector[N_na_s_number] y_imputed;
  real mu_imputed;
  real<lower = 0> sigma_imputed;

  vector[N_spp] spp;
  //real mu_spp;
  //real<lower = 0> sigma_spp;

  vector[N_family] z_fam;
  real mu_fam;
  real<lower = 0> sigma_fam;

  vector[N_order] z_ord;
  real mu_ord;
  real<lower = 0> sigma_ord;

  vector[N_origin] z_ori;
  real mu_ori;
  real<lower = 0> sigma_ori;

  vector[N_status] z_sta;
  real mu_sta;
  real<lower = 0> sigma_sta;

  real<lower = 0> sigma;
```

```
}

transformed parameters{
  vector[N] y_merged;
  y_merged = merge_missing(na_s_number,
                           to_vector(seedsfruit),
                           y_imputed);

  //vector[N_spp] spp;
  //spp = mu_spp + z_spp * sigma_spp;

  vector[N_family] fam;
  fam = mu_fam + z_fam * sigma_fam;

  vector[N_order] ord;
  ord = mu_ord + z_ord * sigma_ord;

  vector[N_origin] ori;
  ori = mu_ori + z_ori * sigma_ori;

  vector[N_status] sta;
  sta = mu_sta + z_sta * sigma_sta;
}

model{
  mu_imputed ~ normal(0, 0.5);
  sigma_imputed ~ exponential(1);
  y_imputed ~ normal(0, 0.5);
  y_merged ~ normal(mu_imputed, sigma_imputed);

  spp ~ normal(0, 1);
  //mu_spp ~ normal(0, 1);
  //sigma_spp ~ exponential(1);

  z_fam ~ normal(0, 0.5);
  mu_fam ~ normal(0, 0.5);
  sigma_fam ~ exponential(1);

  z_ord ~ normal(0, 0.5);
  mu_ord ~ normal(0, 0.5);
  sigma_ord ~ exponential(1);

  z_ori ~ normal(0, 0.5);
  mu_ori ~ normal(0, 0.5);
  sigma_ori ~ exponential(1);

  z_sta ~ normal(0, 0.5);
  mu_sta ~ normal(0, 0.5);
  sigma_sta ~ exponential(1);

  sigma ~ exponential(1);

  for (i in 1:N) {
    y_merged[i] ~ student_t(2, spp[species[i]] +
                           fam[family[i]] +
                           ord[order[i]] +
                           ori[origin[i]] +
                           sta[status[i]], sigma);
  }

}

generated quantities{

  array[N] real ppcheck;
  vector[N] mu;
```

```
      for (i in 1:N) {
        mu[i] = spp[species[i]] +
                fam[family[i]] +
                ord[order[i]] +
                ori[origin[i]] +
                sta[status[i]];
      }

      ppcheck = student_t_rng(2, mu, sigma);

    }

    ')
```

S3.1.4.2 Fitting the model

```
file <- paste0(getwd(), '/seeds_fruit.stan')
fit_seeds_frut <- cmdstan_model(file, compile = T)

mod_seeds_frut <-
  fit_seeds_frut$sample(
    data = dat_plant,
    iter_warmup = 5e3,
    iter_sampling = 10e3,
    thin = 3,
    chains = 3,
    parallel_chains = 3,
    seed = 123
  )
```

S3.1.4.3 Sampling diagnostics

```
sum_seeds_frut <- mod_seeds_frut$summary()

mod_diagnostics(mod_seeds_frut, sum_seeds_frut)
```

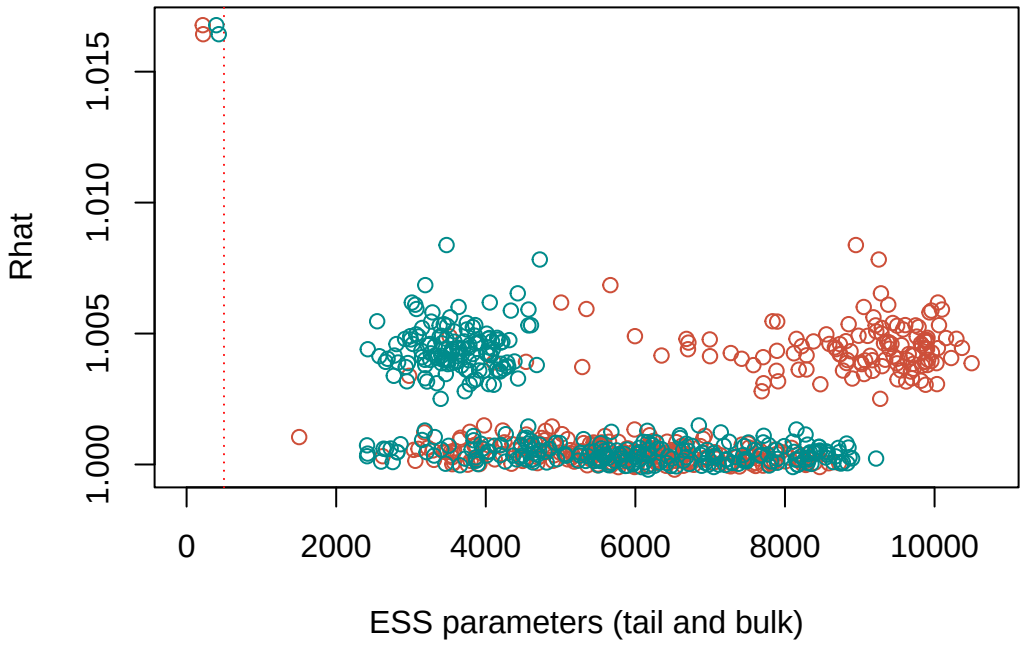


Figure S8: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_seeds_frut <- mod_seeds_frut$draws('ppcheck', format = 'matrix')

plot(density(dat_plant$seedsfruit[which(!is.na(plant_traits$seedsfruit))]),
     xlab = 'Seeds per fruit', xlim = c(-10, 9), ylim = c(0, 0.7),
     main = '')
for (i in 1:100) lines(density(ppcheck_seeds_frut[i, ]))
lines(density(dat_plant$seedsfruit[which(!is.na(plant_traits$seedsfruit))]),
      col = 'red', lwd = 3)
```

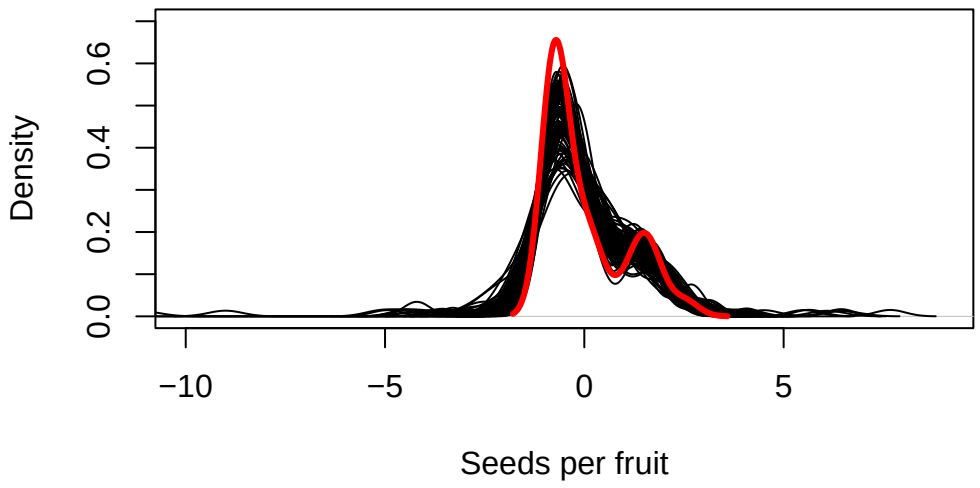


Figure S9: Posterior predictive checks of the model describing seeds per fruit among species.

S3.1.4.4 Extracting posterior draws

```
est_seeds_frut <- extract_post_plants(mod_seeds_frut, dist = 'none')
```

S3.1.4.5 Plotting the posterior distribution

```
rm('ppcheck_seeds_frut')

plot(NULL, xlim = c(0, 70), ylim = c(-3, 4),
     xlab = 'Plant species',
     ylab = 'Seeds per fruit (z-scores)')
segments(y0 = apply(est_seeds_frut, 2, quantile, 0.025),
         y1 = apply(est_seeds_frut, 2, quantile, 0.975),
         x0 = 1:69)
points(1:69, dat_plant$seedsfruit)
points(1:69, apply(est_seeds_frut, 2, mean), pch = 16, col = 'red')
points(dat_plant$na_s_number,
       apply(est_seeds_frut[, dat_plant$na_s_number], 2, mean),
       pch = 16, col = 'cyan4')
```

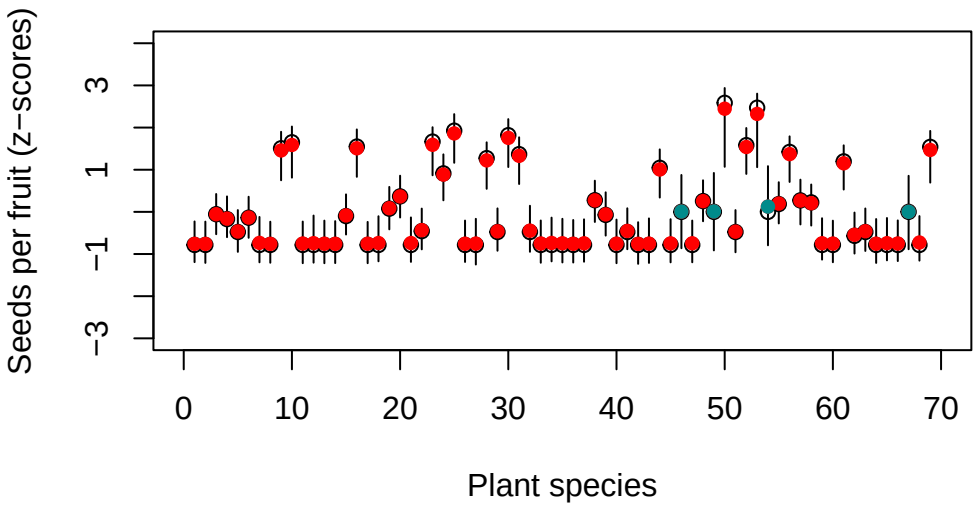


Figure S10: Posterior distributions of seeds per fruit among fruiting species involve in seed dispersal networks in O'ahu island. Error bars indicate 95% CI, red dotes average values and green dotes shows imputed values

S3.1.5 Carbohidrates

S3.1.5.1 Stan code

```
dat_plant$wscarbs[which(!is.na(plant_traits$wscarbs))] <-
  as.vector(scale(dat_plant$wscarbs[which(!is.na(plant_traits$wscarbs))]))
```

```
cat(file = 'carbs.stan',
    '
    functions{
      vector merge_missing(array[] int miss_indxex, vector x_obs, vector x_miss) {
              int N = dims(x_obs)[1];
              int N_miss = dims(x_miss)[1];
              vector[N] merge;
              merge = x_obs;
              for (i in 1:N_miss) {
                  merge[miss_indxex[i]] = x_miss[i];
              }
          return merge;
      }
    }

    data{
      int N;
      int N_spp;
```

```
  int N_family;
  int N_order;
  int N_origin;
  int N_status;
  int N_na_f_width;
  int N_na_s_width;
  int N_na_s_number;
  int N_na_carbs;
  int N_na_lipid;
  int N_na_protein;
  array[N_na_f_width] int na_f_width;
  array[N_na_s_width] int na_s_width;
  array[N_na_s_number] int na_s_number;
  array[N_na_carbs] int na_carbs;
  array[N_na_lipid] int na_lipid;
  array[N_na_protein] int na_protein;
  array[N] int species;
  array[N] int family;
  array[N] int order;
  array[N] int origin;
  array[N] int status;
  vector[N] wscarbs;
}

parameters{

  // imputed wscarbs
  vector[N_na_carbs] y_imputed;
  real mu_imputed;
  real<lower = 0> sigma_imputed;

  vector[N_spp] spp;
  //real mu_spp;
  //real<lower = 0> sigma_spp;

  vector[N_family] z_fam;
  real mu_fam;
  real<lower = 0> sigma_fam;

  vector[N_order] z_ord;
  real mu_ord;
  real<lower = 0> sigma_ord;

  vector[N_origin] z_ori;
  real mu_ori;
  real<lower = 0> sigma_ori;

  vector[N_status] z_sta;
  real mu_sta;
  real<lower = 0> sigma_sta;

  real<lower = 0> sigma;

}

transformed parameters{
  vector[N] y_merged;
  y_merged = merge_missing(na_carbs,
                           to_vector(wscarbs),
                           y_imputed);

  //vector[N_spp] spp;
  //spp = mu_spp + z_spp * sigma_spp;

  vector[N_family] fam;
  fam = mu_fam + z_fam * sigma_fam;
```

```
  vector[N_order] ord;
  ord = mu_ord + z_ord * sigma_ord;

  vector[N_origin] ori;
  ori = mu_ori + z_ori * sigma_ori;

  vector[N_status] sta;
  sta = mu_sta + z_sta * sigma_sta;
}

model{
  mu_imputed ~ normal(0, 0.5);
  sigma_imputed ~ exponential(0.5);
  y_imputed ~ normal(0, 0.5);
  y_merged ~ normal(mu_imputed, sigma_imputed);

  spp ~ normal(0, 0.5);
  //mu_spp ~ normal(0, 1);
  //sigma_spp ~ exponential(1);

  z_fam ~ normal(0, 0.5);
  mu_fam ~ normal(0, 0.5);
  sigma_fam ~ exponential(1);

  z_ord ~ normal(0, 0.5);
  mu_ord ~ normal(0, 0.5);
  sigma_ord ~ exponential(1);

  z_ori ~ normal(0, 0.5);
  mu_ori ~ normal(0, 0.5);
  sigma_ori ~ exponential(1);

  z_sta ~ normal(0, 0.5);
  mu_sta ~ normal(0, 0.5);
  sigma_sta ~ exponential(1);

  sigma ~ exponential(1);

  for (i in 1:N) {
    y_merged[i] ~ student_t(7, spp[species[i]] +
                          fam[family[i]] +
                          ord[order[i]] +
                          ori[origin[i]] +
                          sta[status[i]], sigma);
  }

}

generated quantities{

  array[N] real ppcheck;
  vector[N] mu;

  for (i in 1:N) {
    mu[i] = spp[species[i]] +
            fam[family[i]] +
            ord[order[i]] +
            ori[origin[i]] +
            sta[status[i]];
  }

  ppcheck = student_t_rng(7, mu, sigma);

}

')
```

S3.1.5.2 Fitting the model

```
file <- paste0(getwd(), '/carbs.stan')
fit_carbs <- cmdstan_model(file, compile = T)

mod_carbs <-
  fit_carbs$sample(
    data = dat_plant,
    iter_warmup = 500,
    iter_sampling = 6e3,
    thin = 3,
    chains = 3,
    parallel_chains = 3,
    seed = 555
  )
```

S3.1.5.3 Sampling diagnostics

```
sum_carbs <- mod_carbs$summary()

mod_diagnostics(mod_carbs, sum_carbs)
```

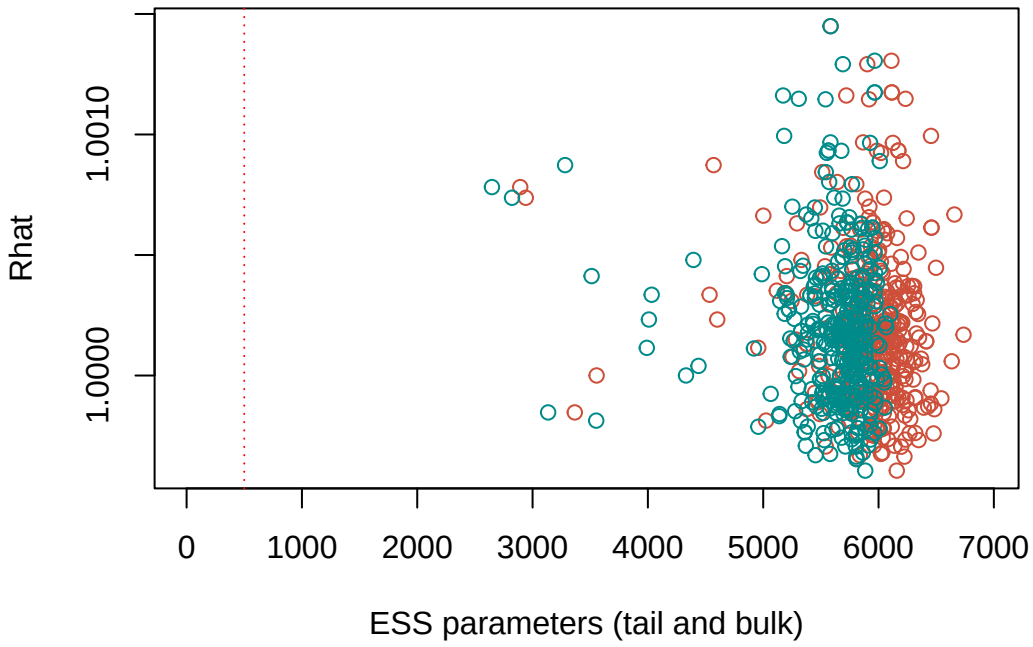


Figure S11: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_carbs <- mod_carbs$draws('ppcheck', format = 'matrix')

plot(density(dat_plant$wscarbs[which(!is.na(plant_traits$wscarbs))]),
     xlab = 'Carbs', xlim = c(-5, 7), ylim = c(0, 1.5), main = '')
for (i in 1:100) lines(density(ppcheck_carbs[i, ]))
lines(density(dat_plant$wscarbs[which(!is.na(plant_traits$wscarbs))]),
      col = 'red', lwd = 3)
```

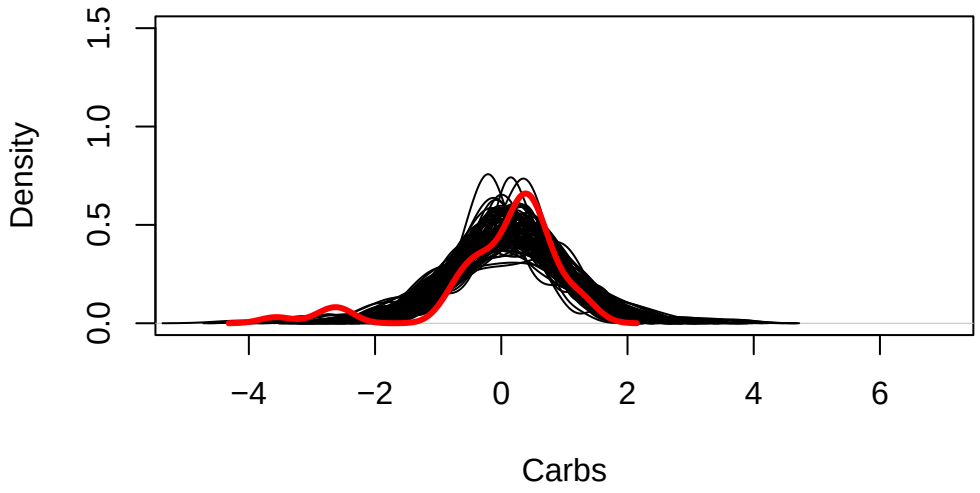


Figure S12: Posterior predictive checks of the model describing carbohidrates content on fruits among species.

S3.1.5.4 Extracting posterior draws

```
est_carbs <- extract_post_plants(mod_carbs, dist = 'none')
```

S3.1.5.5 Plotting the posterior distribution

```
rm('ppcheck_carbs')

plot(NULL, xlim = c(0, 70), ylim = c(-4, 2.5),
     xlab = 'Plant species',
     ylab = 'Carbs (z-scores)')
segments(y0 = apply(est_carbs, 2, quantile, 0.025),
         y1 = apply(est_carbs, 2, quantile, 0.975),
         x0 = 1:69)
points(1:69, dat_plant$wscarbs)
```

```
points(1:69, apply(est_carbs, 2, mean), pch = 16, col = 'red')
points(dat_plant$na_carbs,
       apply(est_carbs[, dat_plant$na_carbs], 2, mean),
       pch = 16, col = 'cyan4')
```

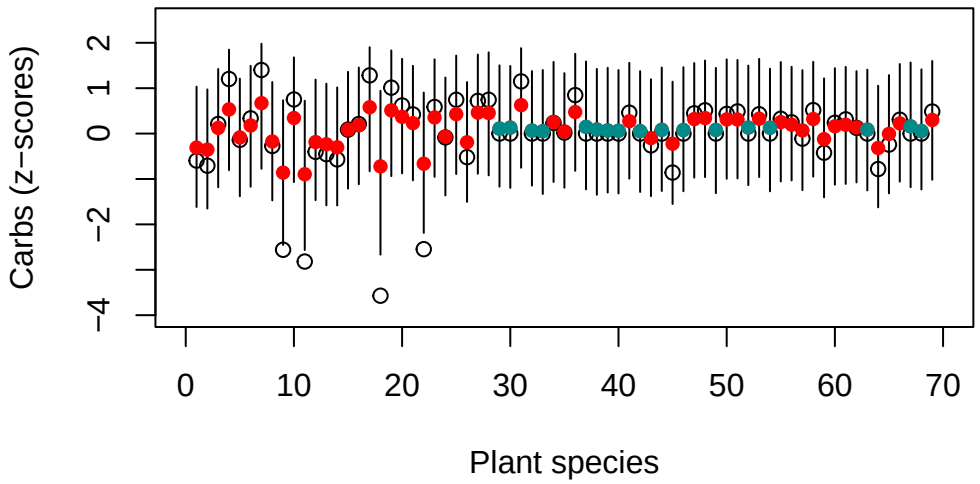


Figure S13: Posterior distributions of carbohidrates content among fruiting species involve in seed dispersal networks in O'ahu island. Error bars indicate 95% CI, red dotes average values and green dotes shows imputed values

S3.1.6 Lipids

S3.1.6.1 Stan code

```
dat_plant$lipid[which(!is.na(plant_traits$lipid))] <-
  as.vector(scale(dat_plant$lipid[which(!is.na(plant_traits$lipid))]))
```

```
cat(file = 'lipids.stan',
    '
    functions{
      vector merge_missing(array[] int miss_indxex, vector x_obs, vector x_miss) {
              int N = dims(x_obs)[1];
              int N_miss = dims(x_miss)[1];
              vector[N] merge;
              merge = x_obs;
              for (i in 1:N_miss) {
                  merge[miss_indxex[i]] = x_miss[i];
```

```
          }
      return merge;
  }
}

data{
  int N;
  int N_spp;
  int N_family;
  int N_order;
  int N_origin;
  int N_status;
  int N_na_f_width;
  int N_na_s_width;
  int N_na_s_number;
  int N_na_carbs;
  int N_na_lipid;
  int N_na_protein;
  array[N_na_f_width] int na_f_width;
  array[N_na_s_width] int na_s_width;
  array[N_na_s_number] int na_s_number;
  array[N_na_carbs] int na_carbs;
  array[N_na_lipid] int na_lipid;
  array[N_na_protein] int na_protein;
  array[N] int species;
  array[N] int family;
  array[N] int order;
  array[N] int origin;
  array[N] int status;
  vector[N] lipid;
}

parameters{

  // imputed lipid
  vector[N_na_lipid] y_imputed;
  real mu_imputed;
  real<lower = 0> sigma_imputed;

  vector[N_spp] spp;
  //real mu_spp;
  //real<lower = 0> sigma_spp;

  vector[N_family] z_fam;
  real mu_fam;
  real<lower = 0> sigma_fam;

  vector[N_order] z_ord;
  real mu_ord;
  real<lower = 0> sigma_ord;

  vector[N_origin] z_ori;
  real mu_ori;
  real<lower = 0> sigma_ori;

  vector[N_status] z_sta;
  real mu_sta;
  real<lower = 0> sigma_sta;

  real<lower = 0> sigma;

}

transformed parameters{
  vector[N] y_merged;
  y_merged = merge_missing(na_lipid,
                           to_vector(lipid),
```

```
                              y_imputed);

  //vector[N_spp] spp;
  //spp = mu_spp + z_spp * sigma_spp;

  vector[N_family] fam;
  fam = mu_fam + z_fam * sigma_fam;

  vector[N_order] ord;
  ord = mu_ord + z_ord * sigma_ord;

  vector[N_origin] ori;
  ori = mu_ori + z_ori * sigma_ori;

  vector[N_status] sta;
  sta = mu_sta + z_sta * sigma_sta;
}

model{
  mu_imputed ~ normal(0, 0.5);
  sigma_imputed ~ exponential(1);
  y_imputed ~ normal(0, 0.5);
  y_merged ~ normal(mu_imputed, sigma_imputed);

  spp ~ normal(0, 0.5);
  //mu_spp ~ normal(0, 1);
  //sigma_spp ~ exponential(1);

  z_fam ~ normal(0, 0.5);
  mu_fam ~ normal(0, 0.5);
  sigma_fam ~ exponential(1);

  z_ord ~ normal(0, 0.5);
  mu_ord ~ normal(0, 0.5);
  sigma_ord ~ exponential(1);

  z_ori ~ normal(0, 0.5);
  mu_ori ~ normal(0, 0.5);
  sigma_ori ~ exponential(1);

  z_sta ~ normal(0, 0.5);
  mu_sta ~ normal(0, 0.5);
  sigma_sta ~ exponential(1);

  sigma ~ exponential(1);

  for (i in 1:N) {
    y_merged[i] ~ normal(spp[species[i]] +
                         fam[family[i]] +
                         ord[order[i]] +
                         ori[origin[i]] +
                         sta[status[i]], sigma);
  }

}

generated quantities{

  array[N] real ppcheck;
  vector[N] mu;

  for (i in 1:N) {
    mu[i] = spp[species[i]] +
            fam[family[i]] +
            ord[order[i]] +
            ori[origin[i]] +
            sta[status[i]];
```

```
      }

      ppcheck = normal_rng(mu, sigma);

    }

    ')
```

S3.1.6.2 Fitting the model

```
file <- paste0(getwd(), '/lipids.stan')
fit_lipid <- cmdstan_model(file, compile = T)

mod_lipid <-
  fit_lipid$sample(
    data = dat_plant,
    iter_warmup = 1e3,
    iter_sampling = 9e3,
    thin = 5,
    chains = 3,
    parallel_chains = 3,
    seed = 555
  )
```

S3.1.6.3 Sampling diagnostics

```
sum_lipid <- mod_lipid$summary()

mod_diagnostics(mod_lipid, sum_lipid)
```

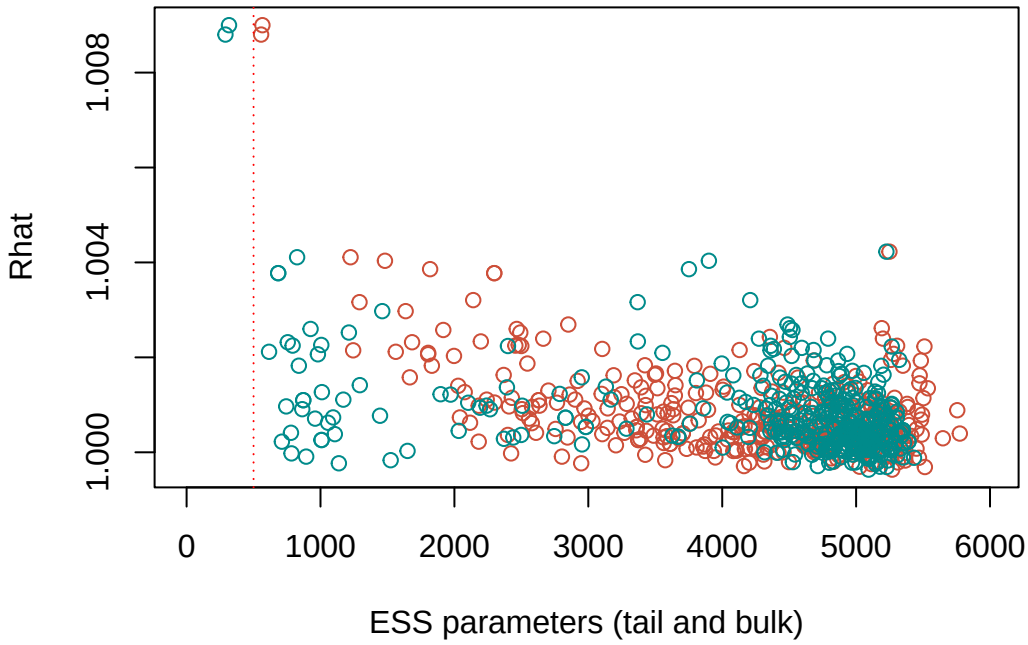


Figure S14: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_lipid <- mod_lipid$draws('ppcheck', format = 'matrix')

plot(density(dat_plant$lipid[which(!is.na(plant_traits$lipid))]),
     xlab = 'lipid', xlim = c(-5, 5), ylim = c(0, 1), main = '')
for (i in 1:100) lines(density(ppcheck_lipid[i, ]))
lines(density(dat_plant$lipid[which(!is.na(plant_traits$lipid))]),
      col = 'red', lwd = 3)
```

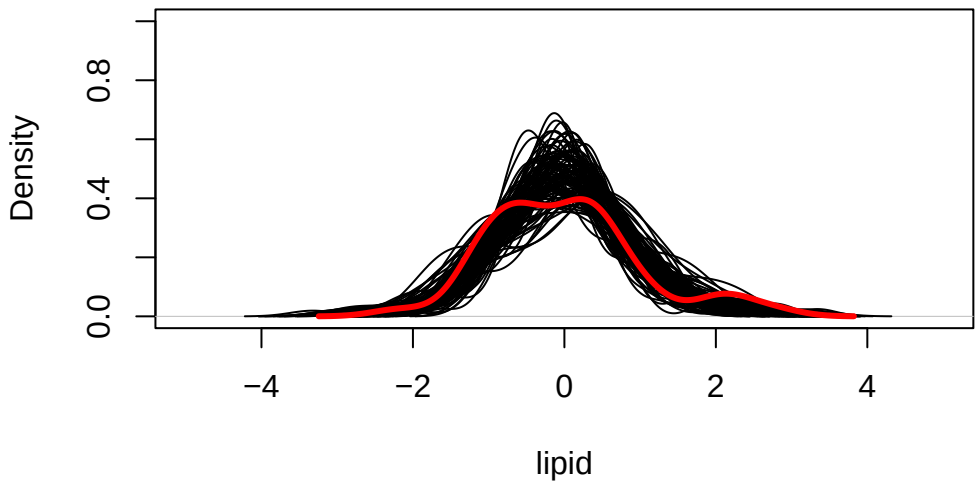


Figure S15: Posterior predictive checks of the model describing lipinds content on fruits among species.

S3.1.6.4 Extracting posterior draws

```
est_lipid <- extract_post_plants(mod_lipid, dist = 'none')
```

S3.1.6.5 Plotting the posterior distribution

```
rm('ppcheck_lipid')

plot(NULL, xlim = c(0, 70), ylim = c(-4, 3.5),
     xlab = 'Plant species',
     ylab = 'lipid (z-scores)')
segments(y0 = apply(est_lipid, 2, quantile, 0.025),
         y1 = apply(est_lipid, 2, quantile, 0.975),
         x0 = 1:69)
points(1:69, dat_plant$lipid)
points(1:69, apply(est_lipid, 2, mean), pch = 16, col = 'red')
points(dat_plant$na_lipid,
       apply(est_lipid[, dat_plant$na_lipid], 2, mean),
       pch = 16, col = 'cyan4')
```

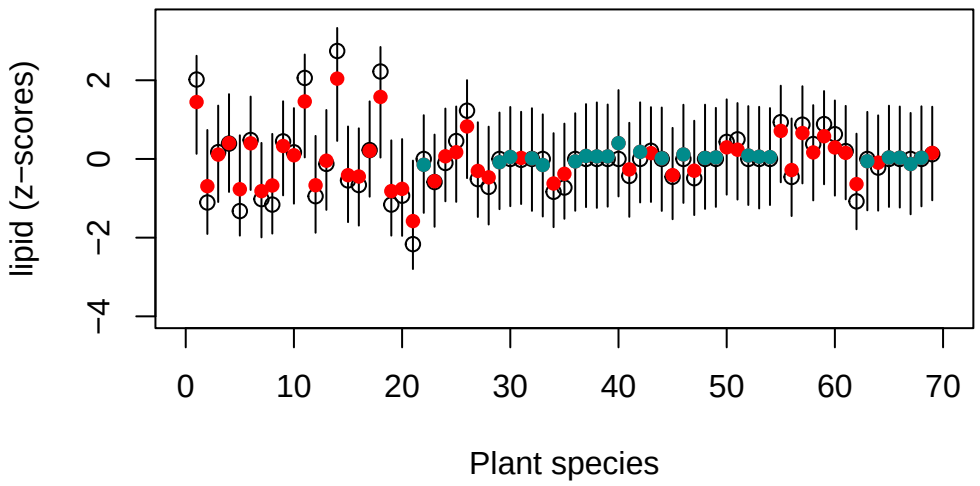


Figure S16: Posterior distributions of lipinds content among fruiting species involve in seed dispersal networks in O'ahu island. Error bars indicate 95% CI, red dotes average values and green dotes shows imputed values

S3.1.7 Protein

S3.1.7.1 Stan code

```
dat_plant$protein[which(!is.na(plant_traits$protein))] <-
   as.vector(scale(dat_plant$protein[which(!is.na(plant_traits$protein))]))
```

```
cat(file = 'protein.stan',
    '
    functions{
      vector merge_missing(array[] int miss_indxex, vector x_obs, vector x_miss) {
              int N = dims(x_obs)[1];
              int N_miss = dims(x_miss)[1];
              vector[N] merge;
              merge = x_obs;
              for (i in 1:N_miss) {
                  merge[miss_indxex[i]] = x_miss[i];
              }
          return merge;
      }
    }

    data{
      int N;
      int N_spp;
```

```
  int N_family;
  int N_order;
  int N_origin;
  int N_status;
  int N_na_f_width;
  int N_na_s_width;
  int N_na_s_number;
  int N_na_carbs;
  int N_na_lipid;
  int N_na_protein;
  array[N_na_f_width] int na_f_width;
  array[N_na_s_width] int na_s_width;
  array[N_na_s_number] int na_s_number;
  array[N_na_carbs] int na_carbs;
  array[N_na_lipid] int na_lipid;
  array[N_na_protein] int na_protein;
  array[N] int species;
  array[N] int family;
  array[N] int order;
  array[N] int origin;
  array[N] int status;
  vector[N] protein;
}

parameters{

  // imputed protein
  vector[N_na_protein] y_imputed;
  real mu_imputed;
  real<lower = 0> sigma_imputed;

  vector[N_spp] spp;
  //real mu_spp;
  //real<lower = 0> sigma_spp;

  vector[N_family] z_fam;
  real mu_fam;
  real<lower = 0> sigma_fam;

  vector[N_order] z_ord;
  real mu_ord;
  real<lower = 0> sigma_ord;

  vector[N_origin] z_ori;
  real mu_ori;
  real<lower = 0> sigma_ori;

  vector[N_status] z_sta;
  real mu_sta;
  real<lower = 0> sigma_sta;

  real<lower = 0> sigma;

}

transformed parameters{
  vector[N] y_merged;
  y_merged = merge_missing(na_protein,
                           to_vector(protein),
                           y_imputed);

  //vector[N_spp] spp;
  //spp = mu_spp + z_spp * sigma_spp;

  vector[N_family] fam;
  fam = mu_fam + z_fam * sigma_fam;
```

```
  vector[N_order] ord;
  ord = mu_ord + z_ord * sigma_ord;

  vector[N_origin] ori;
  ori = mu_ori + z_ori * sigma_ori;

  vector[N_status] sta;
  sta = mu_sta + z_sta * sigma_sta;
}

model{
  mu_imputed ~ normal(0, 0.5);
  sigma_imputed ~ exponential(1);
  y_imputed ~ normal(0, 0.5);
  y_merged ~ normal(mu_imputed, sigma_imputed);

  spp ~ normal(0, 0.5);
  //mu_spp ~ normal(0, 0.5);
  //sigma_spp ~ exponential(1);

  z_fam ~ normal(0, 0.5);
  mu_fam ~ normal(0, 0.5);
  sigma_fam ~ exponential(1);

  z_ord ~ normal(0, 0.5);
  mu_ord ~ normal(0, 0.5);
  sigma_ord ~ exponential(1);

  z_ori ~ normal(0, 0.5);
  mu_ori ~ normal(0, 0.5);
  sigma_ori ~ exponential(1);

  z_sta ~ normal(0, 0.5);
  mu_sta ~ normal(0, 0.5);
  sigma_sta ~ exponential(1);

  sigma ~ exponential(1);

  for (i in 1:N) {
    y_merged[i] ~ student_t(8, spp[species[i]] +
                               fam[family[i]] +
                               ord[order[i]] +
                               ori[origin[i]] +
                               sta[status[i]], sigma);
  }

}

generated quantities{

  array[N] real ppcheck;
  vector[N] mu;

  for (i in 1:N) {
    mu[i] = spp[species[i]] +
            fam[family[i]] +
            ord[order[i]] +
            ori[origin[i]] +
            sta[status[i]];
  }

  ppcheck = student_t_rng(8, mu, sigma);

}

')
```

S3.1.7.2 Fitting the model

```
file <- paste0(getwd(), '/protein.stan')
fit_protein <- cmdstan_model(file, compile = T)

mod_protein <-
  fit_protein$sample(
    data = dat_plant,
    iter_warmup = 1e3,
    iter_sampling = 9e3,
    thin = 5,
    chains = 3,
    parallel_chains = 3,
    seed = 555
  )
```

S3.1.7.3 Sampling diagnostics

```
sum_protein <- mod_protein$summary()

mod_diagnostics(mod_protein, sum_protein)
```

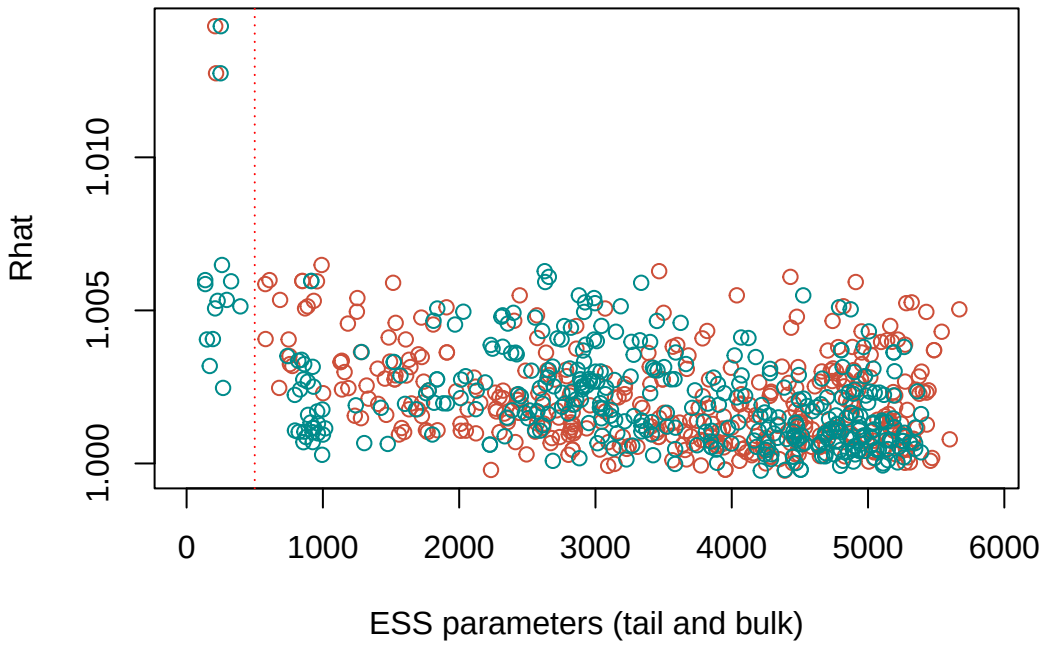


Figure S17: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_protein <- mod_protein$draws('ppcheck', format = 'matrix')

plot(density(dat_plant$protein[which(!is.na(plant_traits$protein))]),
     xlab = 'protein', xlim = c(-5, 5), ylim = c(0, 1), main = '')
for (i in 1:100) lines(density(ppcheck_protein[i, ]))
lines(density(dat_plant$protein[which(!is.na(plant_traits$protein))]),
      col = 'red', lwd = 3)
```

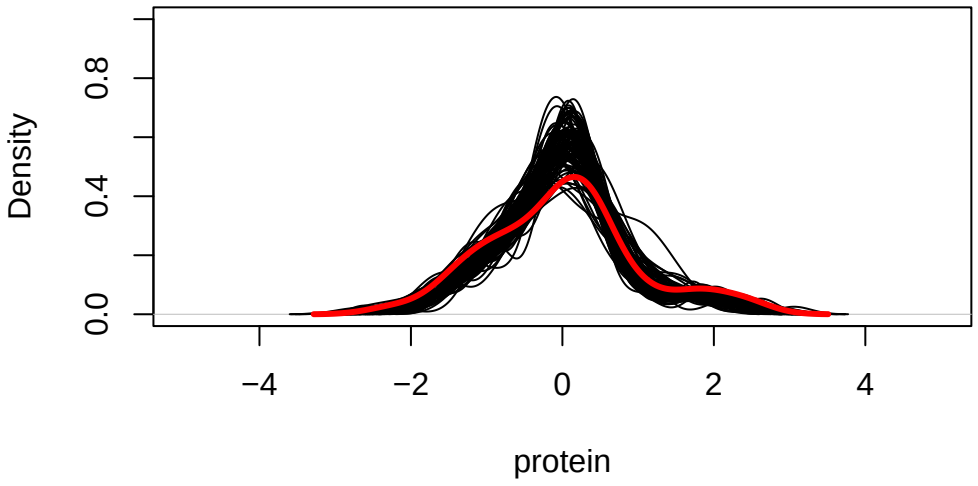


Figure S18: Posterior predictive checks of the model describing protein content on fruits among species.

S3.1.7.4 Extracting posterior draws

```
est_protein <- extract_post_plants(mod_protein, dist = 'none')
```

S3.1.7.5 Plotting the posterior distribution

```
rm('ppcheck_protein')

plot(NULL, xlim = c(0, 70), ylim = c(-3, 3),
     xlab = 'Plant species',
     ylab = 'protein (z-scores)')
segments(y0 = apply(est_protein, 2, quantile, 0.025),
         y1 = apply(est_protein, 2, quantile, 0.975),
         x0 = 1:69)
points(1:69, dat_plant$protein)
```

```
points(1:69, apply(est_protein, 2, mean), pch = 16, col = 'red')
points(dat_plant$na_protein,
       apply(est_protein[, dat_plant$na_protein], 2, mean),
       pch = 16, col = 'cyan4')
```

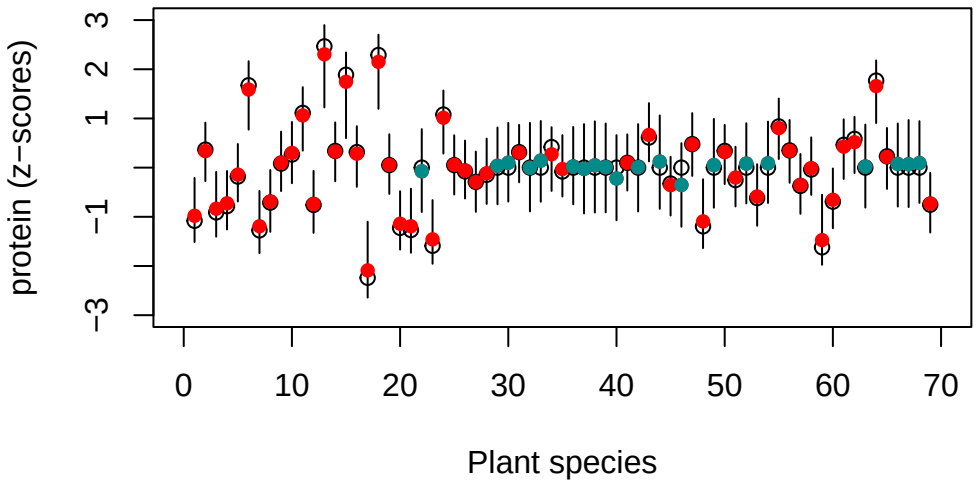


Figure S19: Posterior distributions of protein content among fruiting species involve in seed dispersal networks in O'ahu island. Error bars indicate 95% CI, red dotes average values and green dotes shows imputed values

S3.1.8 Functional space of fruits

```
plants_codes <-
  tibble(code = as.numeric(plant_traits$species),
         spp = plant_traits$species)

plant_functional_traits <-
  lapply(seq_along(plants_codes$code), FUN =
         function(x) {

           i <- plants_codes$code[x]

           fruit_diam <- est_fdiam[, i, drop = T]
           seed_diam <- est_sdiam[, i, drop = T]
           seed_num <- est_seeds_frut[, i, drop = T]
           carbs <- est_carbs[, i, drop = T]
           lipids <- est_lipid[, i, drop = T]
           protein <- est_protein[, i, drop = T]
```

```
            tibble(
              sp = plants_codes$spp[x],
              fruit_diam = fruit_diam,
              seed_diam = seed_diam,
              seed_num = seed_num,
              carbs = carbs,
              lipids = lipids,
              protein = protein
            )
          })

names(plant_functional_traits) <- plants_codes$spp

#saveRDS(plant_functional_traits, 'plant_traits.rds')

plot_NUT <-
  tibble(carbs = as.vector(est_carbs),
         protein = as.vector(est_protein),
         lipids = as.vector(est_lipid),
         spp = rep(plant_traits$species, 2e3))


plot_NUT1 <-
  ggplot() +
  geom_point(data = plot_NUT, aes(carbs, protein),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = plot_NUT,
    geom = "raster",
    aes(carbs, protein, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = plant_functional_traits[[17]],
             aes(carbs, protein),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[18]],
             aes(carbs, protein),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[1]],
             aes(carbs, protein),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(y = 'Protein (z-scores)', x = 'Carbohydrate (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))

plot_NUT2 <-
  ggplot() +
  geom_point(data = plot_NUT, aes(carbs, lipids),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = plot_NUT,
    geom = "raster",
    aes(carbs, lipids, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = plant_functional_traits[[17]],
             aes(carbs, lipids),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[18]],
             aes(carbs, lipids),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[1]],
             aes(carbs, lipids),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(y = 'Lipids (z-scores)', x = 'Carbohydrate (z-scores)') +
```

```
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))


plot_NUT3 <-
  ggplot() +
  geom_point(data = plot_NUT, aes(protein, lipids),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = plot_NUT,
    geom = "raster",
    aes(protein, lipids, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = plant_functional_traits[[17]],
             aes(protein, lipids),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[18]],
             aes(protein, lipids),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[1]],
             aes(protein, lipids),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(x = 'Protein (z-scores)', y = 'Lipids (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))
```

```
plot_grid(plot_grid(plot_NUT1, plot_NUT2, ncol = 2),
          plot_grid(NULL, plot_NUT3, NULL, ncol = 3,
                    rel_widths = c(0.5, 1.3, 0.5)),
          nrow = 2)
```

```
plot_NUT1.1 <-
  ggplot() +
  geom_point(data = do.call('rbind', plant_functional_traits),
             aes(fruit_diam, seed_diam),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = do.call('rbind', plant_functional_traits),
    geom = "raster",
    aes(fruit_diam, seed_diam, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = plant_functional_traits[[17]],
             aes(fruit_diam, seed_diam),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[18]],
             aes(fruit_diam, seed_diam),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[1]],
             aes(fruit_diam, seed_diam),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(x = 'Fruit diameter (z-scores)', y = 'Seed diameter (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))

plot_NUT2.1 <-
  ggplot() +
  geom_point(data = do.call('rbind', plant_functional_traits),
             aes(fruit_diam, seed_num),
             size = 0.1, alpha = 0.2) +
```

```
  stat_density_2d(
    data = do.call('rbind', plant_functional_traits),
    geom = "raster",
    aes(fruit_diam, seed_num, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = plant_functional_traits[[17]],
             aes(fruit_diam, seed_num),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[18]],
             aes(fruit_diam, seed_num),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[1]],
             aes(fruit_diam, seed_num),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(x = 'Fruit diameter (z-scores)', y = 'Seed per fruit (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))


plot_NUT3.1 <-
  ggplot() +
  geom_point(data = do.call('rbind', plant_functional_traits),
             aes(seed_diam, seed_num),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = do.call('rbind', plant_functional_traits),
    geom = "raster",
    aes(seed_diam, seed_num, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = plant_functional_traits[[17]],
             aes(seed_diam, seed_num),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[18]],
             aes(seed_diam, seed_num),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = plant_functional_traits[[1]],
             aes(seed_diam, seed_num),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(x = 'Seed diameter (z-scores)', y = 'Seeds per fruit (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))
```

```
plot_grid(plot_grid(plot_NUT1.1, plot_NUT2.1, ncol = 2),
          plot_grid(NULL, plot_NUT3.1, NULL, ncol = 3,
                    rel_widths = c(0.5, 1.3, 0.5)),
          nrow = 2)
```

S3.2 Bird Traits

Data wrangling operations and formatting for fitting the models.

```
bird_morpho <- read_xlsx('bird_morphology/BirdMorphol_JMG-Master_2019May17.xlsx',
                         sheet = 1, col_names = T)
```

```
Warning: Expecting date in D1525 / R1525C4: got 'NA'
```

```
Warning: Expecting date in D3278 / R3278C4: got 'NA'

Warning: Expecting date in D3287 / R3287C4: got 'NA'

Warning: Expecting date in D3321 / R3321C4: got 'NA'

Warning: Expecting date in D3345 / R3345C4: got 'NA'

Warning: Expecting date in D3421 / R3421C4: got 'NA'

Warning: Expecting date in D3500 / R3500C4: got 'NA'

Warning: Expecting date in D5204 / R5204C4: got 'NA'
```

```
date_birds <- bird_morpho$date

bird_morpho <-
  lapply(seq_along(bird_morpho), FUN =
           function(i) {

             x <- bird_morpho[[i]]

             n <- sum(grep('^([0-9])', x))

             if (n > 0) {
               df <- tibble(v = as.numeric(x))
               colnames(df) <- colnames(bird_morpho)[i]
               df
             } else {
               bird_morpho[, i]
             }

           })
```

```
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion

Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
```

```
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
Warning in eval_tidy(xs[[j]], mask): NAs introduced by coercion
```

```r
bird_morpho <- as_tibble(do.call('cbind', bird_morpho))

bird_morpho$date <- date_birds

bird_morpho <-
  bird_morpho[, c("site", "date", 'net', 'species',
                  "band_number", 'age', 'sex', 'tail',
                  'tarsus', 'wing', "total_culmen",
                  "nares_to_tip", "width", "depth", "gape",
                  "mass")]

bird_morpho <-
  bird_morpho[, c("site", "date", 'net', 'species',
                  "band_number", 'age', 'sex', 'mass', "depth",
                   "gape")]

bird_morpho <- split(bird_morpho, bird_morpho$species)

bird_morpho <- bird_morpho[unlist(lapply(bird_morpho, FUN = function(x) nrow(x) > 0))]


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

bird_morpho <- bird_morpho[bird_morpho$depth < 30,]
```

```
bird_morpho <- na.omit(bird_morpho)

bird_morpho <- bird_morpho[bird_morpho$gape < 30,]

bird_morpho$species <- as.factor(bird_morpho$species)

bird_morpho2 <-
  bird_morpho |>
  group_by(species) |>
  filter(mass > 0) |>
  transmute(gape = log(median(gape)),
            depth = log(median(depth)),
            mass = log(median(mass))) |>
  unique()

birds_names <- read_xlsx('fecal_songbirds/FecalSongbirds_V1.xlsx',
                         sheet = 2, col_names = T)[, c(1:4, 7)]

birds_names$Acronym <- tolower(birds_names$Acronym)

# saveRDS(birds_names, 'birds_code.rds')

bird_morpho2$species %in% birds_names$Acronym
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE
```

```
birds_names <- birds_names[birds_names$Acronym %in% bird_morpho2$species, ]

colnames(birds_names)[ncol(birds_names)] <- 'species'

bird_morpho2 <- full_join(birds_names, bird_morpho2, by = 'species')

dat_bird <-
  lapply(bird_morpho2, FUN =
           function(x) {
             if (is.character(x)) as.numeric(as.factor(x))
             else as.vector(scale(x))
           })

dat_bird$species <- as.numeric(dat_bird$species)
dat_bird$N <- length(dat_bird$species)
dat_bird$N_spp <- max(dat_bird$species)
dat_bird$N_family <- max(dat_bird$family)
dat_bird$N_genus <- max(dat_bird$genus)
dat_bird$N_order <- max(dat_bird$order)

birds_codes <-
  tibble(spp_code = dat_bird$species,
       spp_name = bird_morpho2$species,
       spp_acronym = bird_morpho2$species,
       ord_code = dat_bird$order,
       ord_name = bird_morpho2$order,
       family_code = dat_bird$family,
       family_name = bird_morpho2$family,
       genus_code = dat_bird$genus,
       genus_name = bird_morpho2$genus)
```

S3.2.1 Mathematical notation of the models

These models estimate species-specific variation in **bill gape**, **bill gape**, and **body mass** among birds participating in seed dispersal networks, while accounting for taxonomic effects (family and order). All models share the same structure; thus, the following mathematical notation applies to each.

$$
\begin{aligned}
y_i &\sim \mathcal{N}(\mu_i, \sigma) \\
\mu_i &= \alpha_{sp} + \theta_{genus} + \phi_{family} + \delta_{order} \\
\alpha_{sp} &\sim \mathcal{N}(0, 1) \\
\sigma &\sim Exp(2) \\
\theta_{genus} &= \mu_\theta + Z_\theta + \sigma_\theta \\
\phi_{family} &= \mu_\phi + Z_\phi + \sigma_\phi \\
\delta_{order} &= \mu_\delta + Z_\delta + \sigma_\delta \\
\mu_{\theta,\phi,\delta} &\sim \mathcal{N}(0, 1) \\
Z_{\theta,\phi,\delta} &\sim \mathcal{N}(0, 1) \\
\sigma_{\theta,\phi,\delta} &\sim Exp(1)
\end{aligned}
$$

S3.2.2 Bill gape

S3.2.2.1 Stan code

```
cat(file = 'gape_model.stan',
    "
    data {

      int N;
      int N_spp;
      int N_genus;
      int N_family;
      int N_order;
      vector[N] gape;
      array[N] int species;
      array[N] int order;
      array[N] int family;
      array[N] int genus;
    }

    parameters {
      vector[N_spp] alpha;
      //real mu_alpha;
      //real<lower = 0> sigma_alpha;
      real<lower = 0> sigma;

      vector[N_order] z_ord;
      real mu_ord;
      real<lower = 0> sigma_ord;

      vector[N_family] z_fam;
```

```
      real mu_fam;
      real<lower = 0> sigma_fam;

      vector[N_genus] z_gen;
      real mu_gen;
      real<lower = 0> sigma_gen;
    }

    transformed parameters {
      vector[N_order] ord;
      ord = mu_ord + z_ord * sigma_ord;

      vector[N_family] fam;
      fam = mu_fam + z_fam * sigma_fam;

      vector[N_genus] gen;
      gen = mu_gen + z_gen * sigma_gen;
    }

    model {

      z_ord ~ normal(0, 1);
      mu_ord ~ normal(0, 1);
      sigma_ord ~ exponential(1);

      z_fam ~ normal(0, 1);
      mu_fam ~ normal(0, 1);
      sigma_fam ~ exponential(1);

      z_gen ~ normal(0, 1);
      mu_gen ~ normal(0, 1);
      sigma_gen ~ exponential(1);

      alpha ~ normal(0, 1);
      sigma ~ exponential(2);

      gape ~ normal(alpha[species] +
                    ord[order] +
                    fam[family] +
                    gen[genus], sigma);
    }

    generated quantities {
      array[N] real ppcheck;

      ppcheck = normal_rng(alpha[species] +
                           ord[order] +
                           fam[family] +
                           gen[genus], sigma);
    }
    ")
```

S3.2.2.2 Fitting the model

```
file <- paste0(getwd(), '/gape_model.stan')
fit_gape <- cmdstan_model(file, compile = T)

mod_gape <-
  fit_gape$sample(
    data = dat_bird,
    chains = 3,
    parallel_chains = 3,
    iter_warmup = 2e3,
```

```
    iter_sampling = 3e4,
    thin = 3,
    seed = 123
  )
```

S3.2.2.3 Sampling diagnostics

```
sum_gape <- mod_gape$summary()

mod_diagnostics(mod_gape, sum_gape)
```

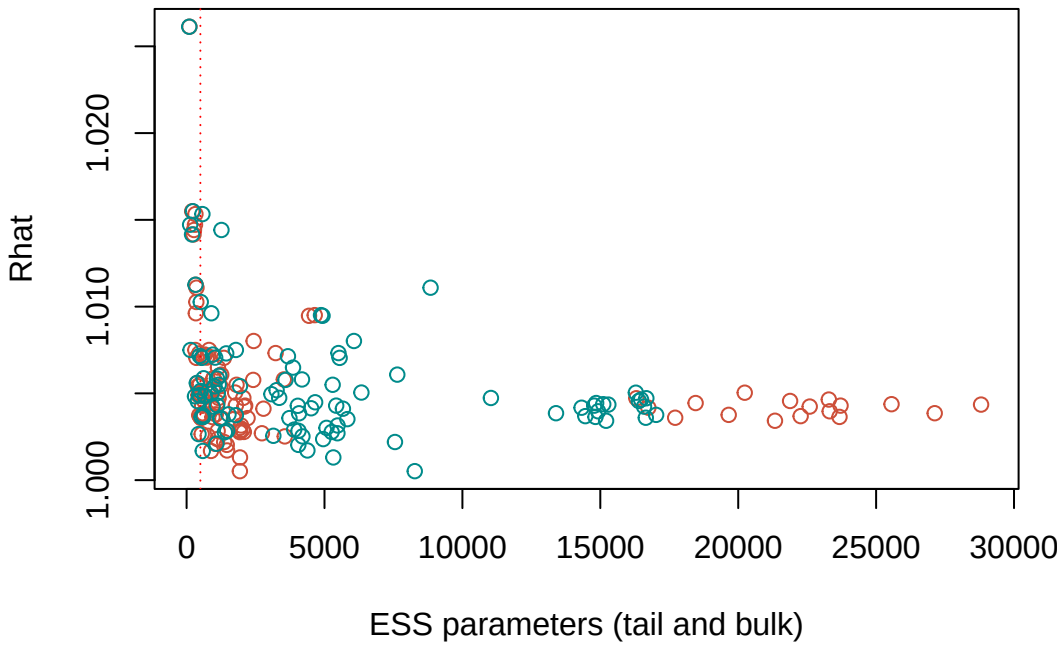


Figure S20: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_gape <- mod_gape$draws('ppcheck', format = 'matrix')

plot(density(dat_bird$gape), xlim = c(-3.5, 3.5), ylim = c(0, 0.6),
     xlab = 'Bill gape', main = '')
for (i in 1:100) lines(density(ppcheck_gape[i, ]), lwd = 0.1)
lines(density(dat_bird$gape), col = 'red', lwd = 2)
```

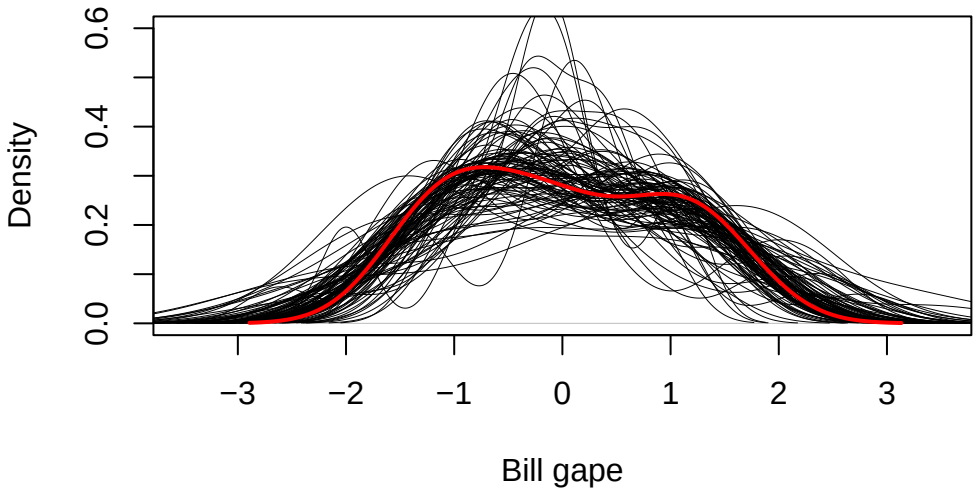


Figure S21: Posterior predictive checks of the model describing bill gape among species.

S3.2.2.4 Extracting posterior draws

```
post_gape <- mod_gape$draws(c('alpha', 'ord', 'fam', 'gen', 'sigma'),
                            format = 'df')

post_gape <-
  lapply(c('alpha', 'ord', 'fam', 'gen', 'sigma'), FUN =
           function(x) {

             post_gape[, grep(x, colnames(post_gape))]

           })

names(post_gape) <- c('alpha', 'ord', 'fam', 'gen', 'sigma')

ppcheck_gape <-
  lapply(seq_along(dat_bird$species), FUN =
           function(x) {

             spp <- dat_bird$species[x]
             order <- dat_bird$order[x]
             family <- dat_bird$family[x]
             genus <- dat_bird$genus[x]

             mu <-
               with(post_gape,
                    {
                      alpha[, spp, drop = T] +
                        fam[, family, drop = T] +
                        gen[, genus, drop = T] +
                        ord[, order, drop = T]
```

```
                    })

            mu <- rnorm(2e3, mu, post_gape$sigma$sigma)

            d <- tibble(x = mu)
            colnames(d) <- paste('var', spp, sep = '_')
            d

          })

ppcheck_gape <- do.call('cbind', ppcheck_gape)

post_gape <-
  lapply(seq_along(dat_bird$species), FUN =
           function(x) {

            spp <- dat_bird$species[x]
            order <- dat_bird$order[x]
            family <- dat_bird$family[x]
            genus <- dat_bird$genus[x]

            mu <-
              with(post_gape,
                   {
                     alpha[, spp, drop = T] +
                       fam[, family, drop = T] +
                       gen[, genus, drop = T] +
                       ord[, order, drop = T]
                   })

            q1 <- quantile(mu, 0.0025) # 99.5% of the posterior distribution
            q2 <- quantile(mu, 0.9975)
            mu <- mu[mu >= q1 & mu <= q2]

            d <- tibble(x = mu[1:2e3])
            colnames(d) <- paste('var', spp, sep = '_')
            d

           })

post_gape <- as_tibble(do.call('cbind', post_gape))

names(post_gape) <- birds_codes$spp_acronym

colnames(ppcheck_gape) <- colnames(post_gape)
```

S3.2.2.5 Plotting the posterior distribution

```
par(mfrow = c(3, 3), mar = c(4, 4, 1.5, 1))
for (i in bird_morpho2$species) {
  plot(NULL, col = 'red',
       xlim = c(-3.5, 3.5), ylim = c(0, 2),
       main = i, xlab = 'Gape (z_scores)', ylab = 'Density')
  abline(v = (log(bird_morpho[bird_morpho$species == i, ]$gape) -
                mean(bird_morpho2$gape)) / sd(bird_morpho2$gape),
         lty = 3, lwd = 0.5)
  # abline(v = dat_bird$gape[which(bird_morpho2$species == i)],
  #        lty = 1.5, lwd = 0.5, col = 'red')
  lines(density(post_gape[, i, drop = T]), col = 'red', lwd = 2)
  #lines(density(ppcheck_gape[, i, drop = T]), lwd = 2)

}
```

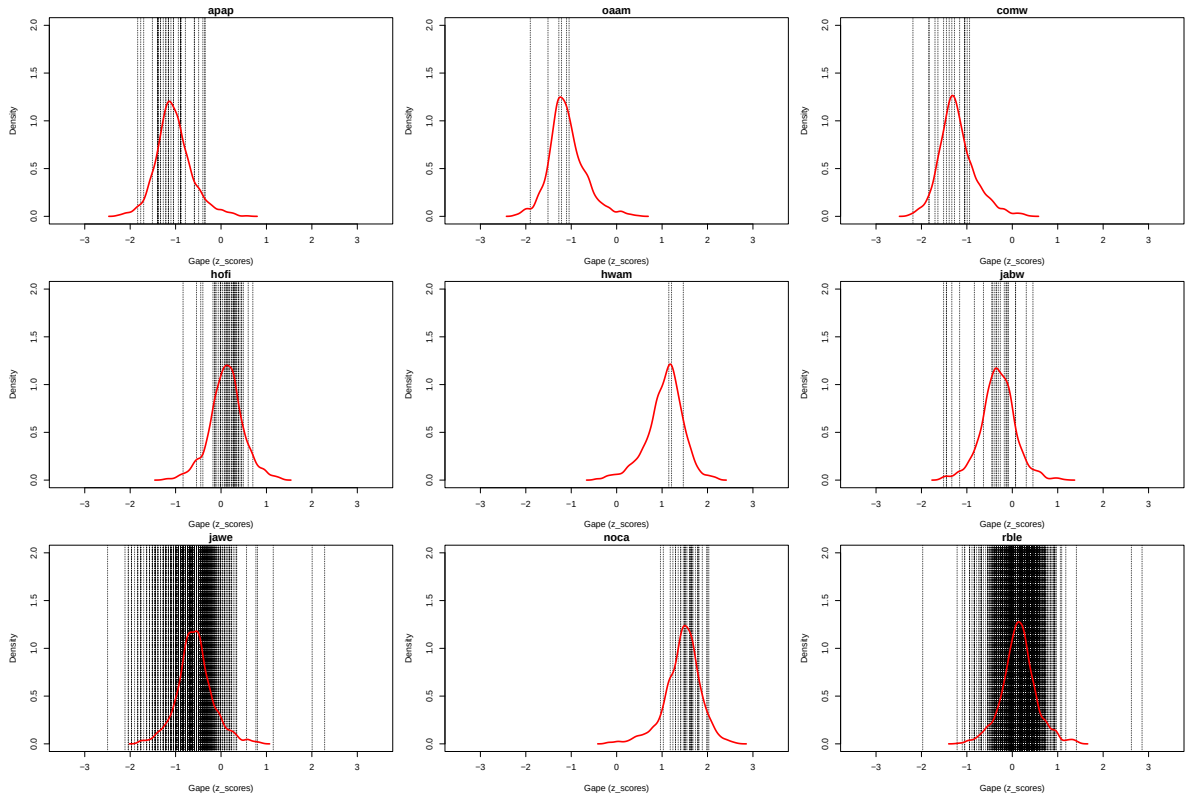


Figure S22: Posterior distributions of the bill gape of species involve in seed dispersal networks in O'ahu island. Each panel belongs to one species; red density line denotes the estimated posterior distribution; vertical dotted line shows observed values that were not used to fit the model

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

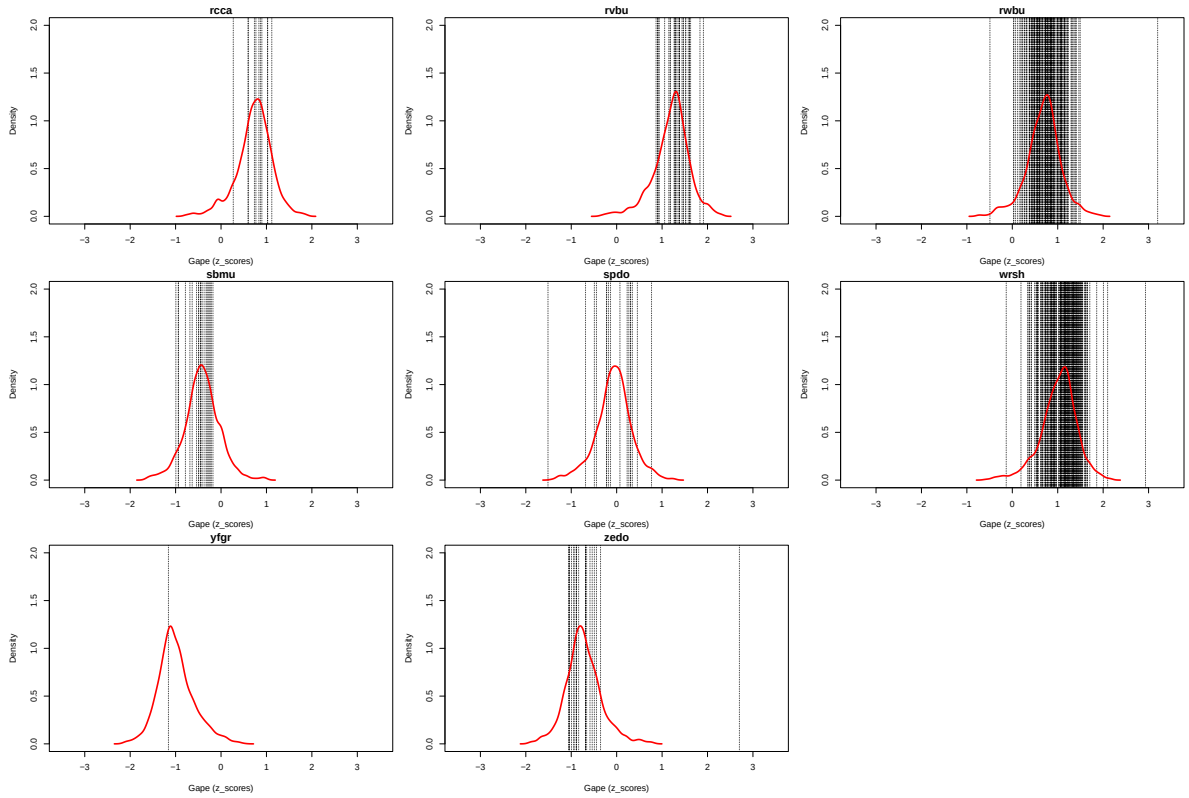


Figure S23: Posterior distributions of the bill gape of species involve in seed dispersal networks in O'ahu island. Each panel belongs to one species; red density line denotes the estimated posterior distribution; vertical dotted line shows observed values that were not used to fit the model

S3.2.3 Bill Depth

S3.2.3.1 Stan code

```
cat(file = 'depth_model.stan',
    "
    data {

      int N;
      int N_spp;
      int N_genus;
      int N_family;
      int N_order;
      vector[N] depth;
      array[N] int species;
      array[N] int order;
      array[N] int family;
      array[N] int genus;
    }
```

```
  parameters {
    vector[N_spp] alpha;
    //real mu_alpha;
    //real<lower = 0> sigma_alpha;
    real<lower = 0> sigma;

    vector[N_order] z_ord;
    real mu_ord;
    real<lower = 0> sigma_ord;

    vector[N_family] z_fam;
    real mu_fam;
    real<lower = 0> sigma_fam;

    vector[N_genus] z_gen;
    real mu_gen;
    real<lower = 0> sigma_gen;
  }

  transformed parameters {
    vector[N_order] ord;
    ord = mu_ord + z_ord * sigma_ord;

    vector[N_family] fam;
    fam = mu_fam + z_fam * sigma_fam;

    vector[N_genus] gen;
    gen = mu_gen + z_gen * sigma_gen;
  }

  model {

    z_ord ~ normal(0, 0.5);
    mu_ord ~ normal(0, 1);
    sigma_ord ~ exponential(1);

    z_fam ~ normal(0, 0.5);
    mu_fam ~ normal(0, 1);
    sigma_fam ~ exponential(1);

    z_gen ~ normal(0, 0.5);
    mu_gen ~ normal(0, 1);
    sigma_gen ~ exponential(1);

    alpha ~ normal(0, 1);
    sigma ~ exponential(0.5);

    depth ~ normal(alpha[species] +
                   ord[order] +
                   fam[family] +
                   gen[genus], sigma);
  }

  generated quantities {
    array[N] real ppcheck;

    ppcheck = normal_rng(alpha[species] +
                         ord[order] +
                         fam[family] +
                         gen[genus], sigma);
  }
  ")
```

S3.2.3.2 Fitting the model

```
file <- paste0(getwd(), '/depth_model.stan')
fit_depth <- cmdstan_model(file, compile = T)

mod_depth <-
  fit_depth$sample(
    data = dat_bird,
    chains = 3,
    parallel_chains = 3,
    iter_warmup = 2e3,
    iter_sampling = 3e4,
    thin = 3,
    seed = 123
  )
```

S3.2.3.3 Sampling diagnostics

```
sum_depth <- mod_depth$summary()

mod_diagnostics(mod_depth, sum_depth)
```

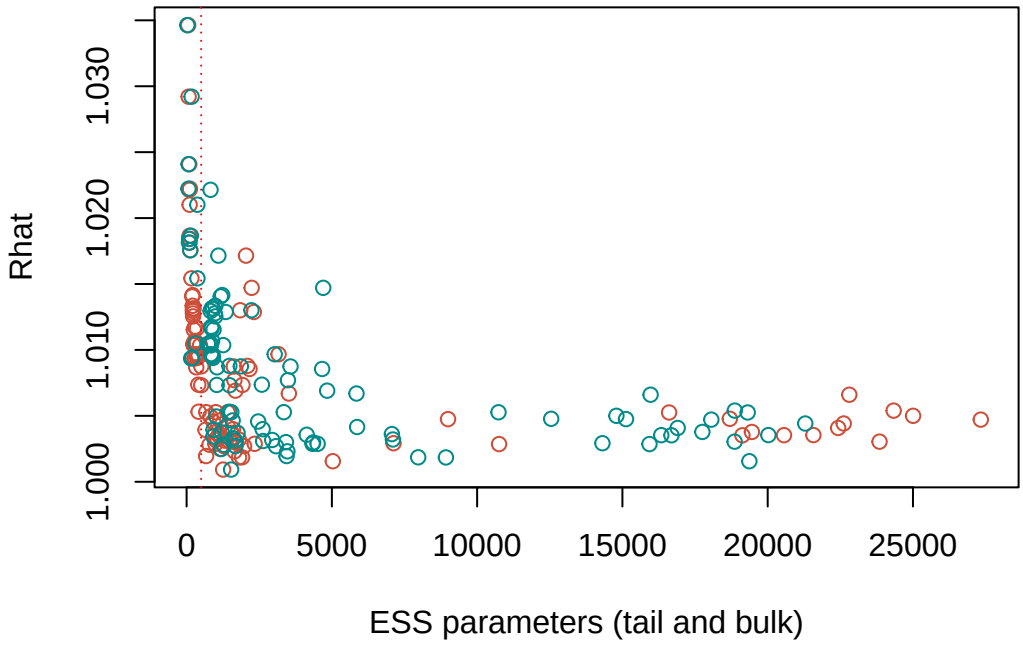


Figure S24: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_depth <- mod_depth$draws('ppcheck', format = 'matrix')

plot(density(dat_bird$depth), xlim = c(-3.5, 3.5), ylim = c(0, 0.6),
```

```
     xlab = 'Bill depth', main = '')
for (i in 1:100) lines(density(ppcheck_depth[i, ]), lwd = 0.1)
lines(density(dat_bird$depth), col = 'red', lwd = 2)
```

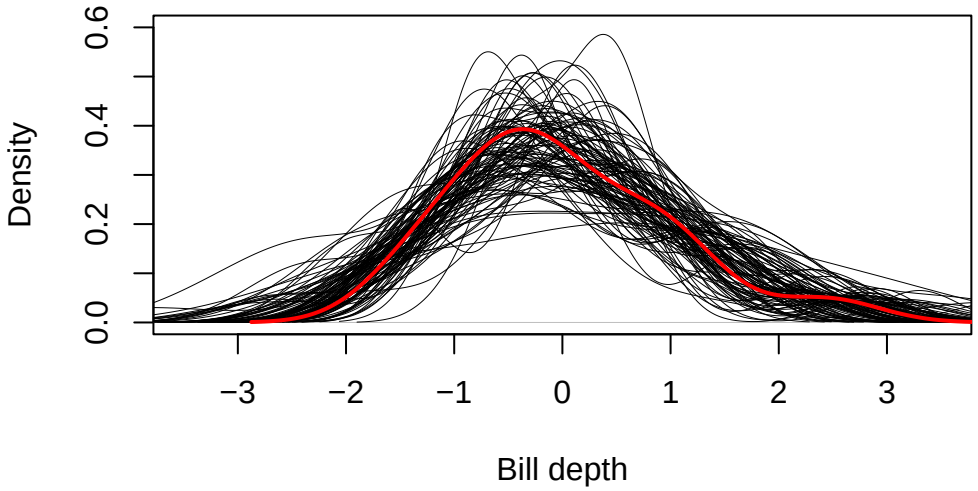


Figure S25: Posterior predictive checks of the model describing bill depth among species.

S3.2.3.4 Extracting posterior draws

```
post_depth <- mod_depth$draws(c('alpha', 'ord', 'fam', 'gen', 'sigma'),
                              format = 'df')

post_depth <-
  lapply(c('alpha', 'ord', 'fam', 'gen', 'sigma'), FUN =
           function(x) {

             post_depth[, grep(x, colnames(post_depth))]

           })

names(post_depth) <- c('alpha', 'ord', 'fam', 'gen', 'sigma')

ppcheck_depth <-
  lapply(seq_along(dat_bird$species), FUN =
           function(x) {

             spp <- dat_bird$species[x]
             order <- dat_bird$order[x]
             family <- dat_bird$family[x]
             genus <- dat_bird$genus[x]
```

```
            mu <-
              with(post_depth,
                   {
                     alpha[, spp, drop = T] +
                       fam[, family, drop = T] +
                       gen[, genus, drop = T] +
                       ord[, order, drop = T]
                   })

            mu <- rnorm(2e3, mu, post_depth$sigma$sigma)

            d <- tibble(x = mu)
            colnames(d) <- paste('var', spp, sep = '_')
            d

          })

ppcheck_depth <- do.call('cbind', ppcheck_depth)

post_depth <-
  lapply(seq_along(dat_bird$species), FUN =
           function(x) {

             spp <- dat_bird$species[x]
             order <- dat_bird$order[x]
             family <- dat_bird$family[x]
             genus <- dat_bird$genus[x]

             mu <-
               with(post_depth,
                    {
                      alpha[, spp, drop = T] +
                        fam[, family, drop = T] +
                        gen[, genus, drop = T] +
                        ord[, order, drop = T]
                    })
             q1 <- quantile(mu, 0.0025)
             q2 <- quantile(mu, 0.9975)
             mu <- mu[mu >= q1 & mu <= q2]
             # set.seed(5)
             # d <- tibble(x = sample(mu, 2e3))
             d <- tibble(x = mu[1:2e3])
             colnames(d) <- paste('var', spp, sep = '_')
             d

           })

post_depth <- as_tibble(do.call('cbind', post_depth))

names(post_depth) <- birds_codes$spp_acronym

colnames(ppcheck_depth) <- colnames(post_depth)
```

S3.2.3.5 Plotting the posterior distribution

```
par(mfrow = c(3, 3), mar = c(4, 4, 1.5, 1))
for (i in bird_morpho2$species) {
  plot(NULL, col = 'red',
       xlim = c(-3.5, 3.5), ylim = c(0, 2),
       main = i, xlab = 'depth (z_scores)', ylab = 'Density')
  abline(v = (log(bird_morpho[bird_morpho$species == i, ]$depth) -
                mean(bird_morpho2$depth)) / sd(bird_morpho2$depth),
         lty = 3, lwd = 0.5)
```

```
  # abline(v = dat_bird$depth[which(bird_morpho2$species == i)],
  #        lty = 1.5, lwd = 0.5, col = 'red')
  lines(density(post_depth[, i, drop = T]), col = 'red', lwd = 2)
  #lines(density(ppcheck_depth[, i, drop = T]), lwd = 2)

}
```

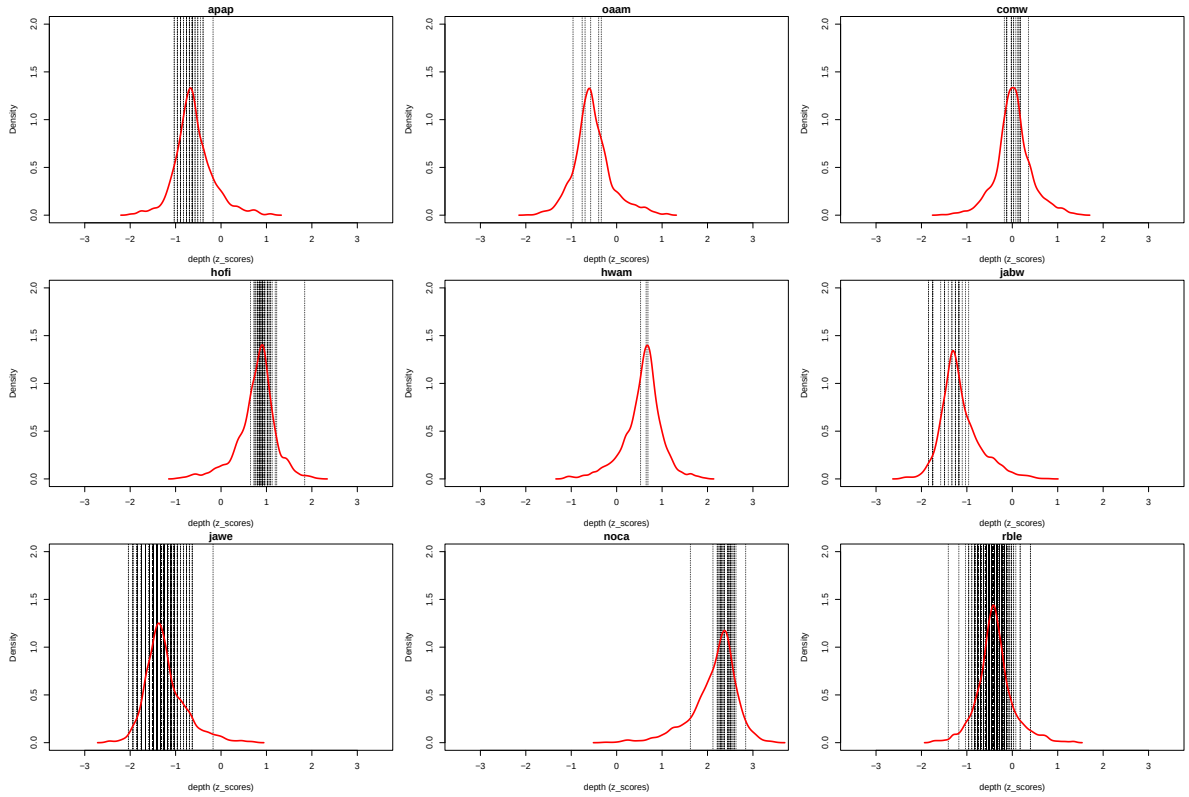


Figure S26: Posterior distributions of the bill depth of species involve in seed dispersal networks in O'ahu island. Each panel belongs to one species; red density line denotes the estimated posterior distribution; vertical dotted line shows observed values that were not used to fit the model

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

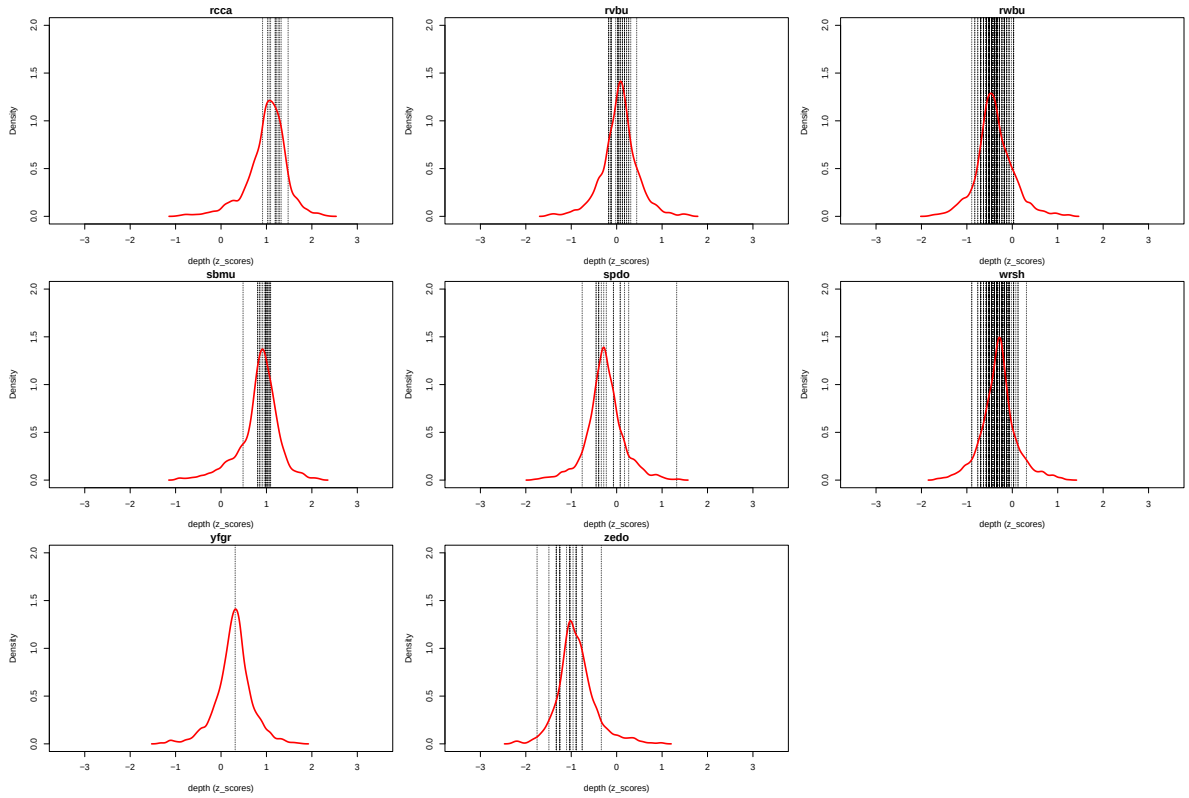


Figure S27: Posterior distributions of the bill depth of species involve in seed dispersal networks in O'ahu island. Each panel belongs to one species; red density line denotes the estimated posterior distribution; vertical dotted line shows observed values that were not used to fit the model

S3.2.4 Body mass

S3.2.4.1 Stan code

```
cat(file = 'mass_model.stan',
    "
    data {

      int N;
      int N_spp;
      int N_genus;
      int N_family;
      int N_order;
      vector[N] mass;
      array[N] int species;
      array[N] int order;
      array[N] int family;
      array[N] int genus;
    }
```

```
  parameters {
    vector[N_spp] alpha;
    //real mu_alpha;
    //real<lower = 0> sigma_alpha;
    real<lower = 0> sigma;

    vector[N_order] z_ord;
    real mu_ord;
    real<lower = 0> sigma_ord;

    vector[N_family] z_fam;
    real mu_fam;
    real<lower = 0> sigma_fam;

    vector[N_genus] z_gen;
    real mu_gen;
    real<lower = 0> sigma_gen;
  }

  transformed parameters {
    vector[N_order] ord;
    ord = mu_ord + z_ord * sigma_ord;

    vector[N_family] fam;
    fam = mu_fam + z_fam * sigma_fam;

    vector[N_genus] gen;
    gen = mu_gen + z_gen * sigma_gen;
  }

  model {

    z_ord ~ normal(0, 0.5);
    mu_ord ~ normal(0, 0.5);
    sigma_ord ~ exponential(1);

    z_fam ~ normal(0, 0.5);
    mu_fam ~ normal(0, 0.5);
    sigma_fam ~ exponential(1);

    z_gen ~ normal(0, 0.5);
    mu_gen ~ normal(0, 0.5);
    sigma_gen ~ exponential(1);

    alpha ~ normal(0, 1);
    sigma ~ exponential(0.5);

    mass ~ normal(alpha[species] +
                  ord[order] +
                  fam[family] +
                  gen[genus], sigma);
  }

  generated quantities {
    array[N] real ppcheck;

    ppcheck = normal_rng(alpha[species] +
                         ord[order] +
                         fam[family] +
                         gen[genus], sigma);
  }
  ")
```

S3.2.4.2 Fitting the model

```
file <- paste0(getwd(), '/mass_model.stan')
fit_mass <- cmdstan_model(file, compile = T)

mod_mass <-
  fit_mass$sample(
    data = dat_bird,
    chains = 3,
    parallel_chains = 3,
    iter_warmup = 2e3,
    iter_sampling = 3e4,
    thin = 3,
    seed = 123
  )
```

S3.2.4.3 Sampling diagnostics

```
sum_mass <- mod_mass$summary()

mod_diagnostics(mod_mass, sum_mass)
```

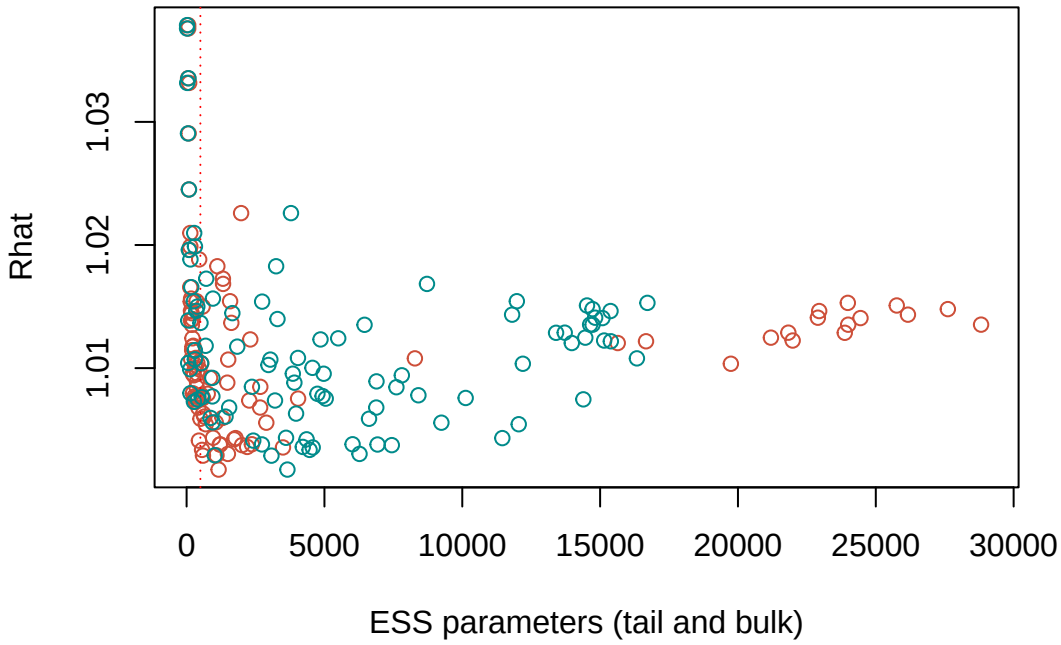


Figure S28: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_mass <- mod_mass$draws('ppcheck', format = 'matrix')

plot(density(dat_bird$mass), xlim = c(-3.5, 3.5), ylim = c(0, 0.6),
```

```
     xlab = 'Body mass', main = '')
for (i in 1:100) lines(density(ppcheck_mass[i, ]), lwd = 0.1)
lines(density(dat_bird$mass), col = 'red', lwd = 2)
```

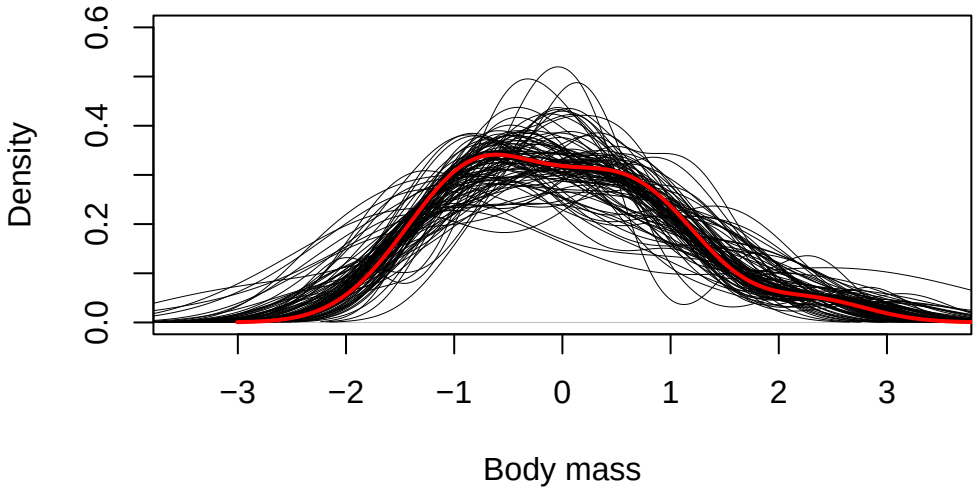


Figure S29: Posterior predictive checks of the model describing body mass among species.

S3.2.4.4 Extracting posterior draws

```
post_mass <- mod_mass$draws(c('alpha', 'ord', 'fam', 'gen', 'sigma'),
                            format = 'df')

post_mass <-
  lapply(c('alpha', 'ord', 'fam', 'gen', 'sigma'), FUN =
           function(x) {

             post_mass[, grep(x, colnames(post_mass))]

           })

names(post_mass) <- c('alpha', 'ord', 'fam', 'gen', 'sigma')

ppcheck_mass <-
  lapply(seq_along(dat_bird$species), FUN =
           function(x) {

             spp <- dat_bird$species[x]
             order <- dat_bird$order[x]
             family <- dat_bird$family[x]
             genus <- dat_bird$genus[x]
```

```
            mu <-
              with(post_mass,
                   {
                     alpha[, spp, drop = T] +
                       fam[, family, drop = T] +
                       gen[, genus, drop = T] +
                       ord[, order, drop = T]
                   })

            mu <- rnorm(2e3, mu, post_mass$sigma$sigma)

            d <- tibble(x = mu)
            colnames(d) <- paste('var', spp, sep = '_')
            d

          })

ppcheck_mass <- do.call('cbind', ppcheck_mass)

post_mass <-
  lapply(seq_along(dat_bird$species), FUN =
           function(x) {

             spp <- dat_bird$species[x]
             order <- dat_bird$order[x]
             family <- dat_bird$family[x]
             genus <- dat_bird$genus[x]

             mu <-
               with(post_mass,
                    {
                      alpha[, spp, drop = T] +
                        fam[, family, drop = T] +
                        gen[, genus, drop = T] +
                        ord[, order, drop = T]
                    })
             q1 <- quantile(mu, 0.0025)
             q2 <- quantile(mu, 0.9975)
             mu <- mu[mu >= q1 & mu <= q2]
             # set.seed(5)
             # d <- tibble(x = sample(mu, 2e3))
             d <- tibble(x = mu[1:2e3])
             colnames(d) <- paste('var', spp, sep = '_')
             d

           })

post_mass <- as_tibble(do.call('cbind', post_mass))

names(post_mass) <- birds_codes$spp_acronym

colnames(ppcheck_mass) <- colnames(post_mass)

bird_morpho <- bird_morpho[bird_morpho$mass >= 0, ]
```

S3.2.4.5 Plotting the posterior distribution

```
par(mfrow = c(3, 3), mar = c(4, 4, 1.5, 1))
for (i in bird_morpho2$species) {
  plot(NULL, col = 'red',
       xlim = c(-3.5, 3.5), ylim = c(0, 2),
       main = i, xlab = 'mass (z_scores)', ylab = 'Density')
  abline(v = (log(bird_morpho[bird_morpho$species == i, ]$mass) -
```

```
                mean(bird_morpho2$mass)) / sd(bird_morpho2$mass),
        lty = 3, lwd = 0.5)
  # abline(v = dat_bird$mass[which(bird_morpho2$species == i)],
  #        lty = 1.5, lwd = 0.5, col = 'red')
  lines(density(post_mass[, i, drop = T]), col = 'red', lwd = 2)
  #lines(density(ppcheck_mass[, i, drop = T]), lwd = 2)

}
```

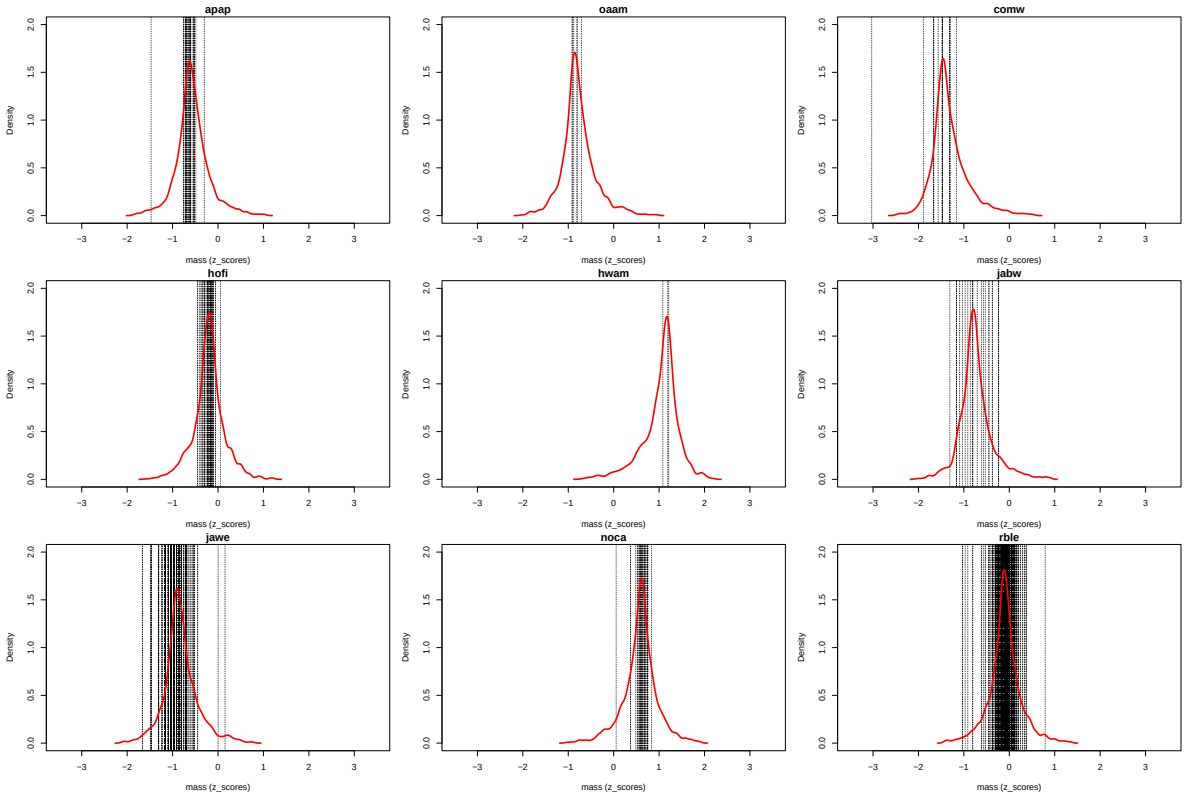


Figure S30: Posterior distributions of the body mass of species involve in seed dispersal networks in O'ahu island. Each panel belongs to one species; red density line denotes the estimated posterior distribution; vertical dotted line shows observed values that were not used to fit the model

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

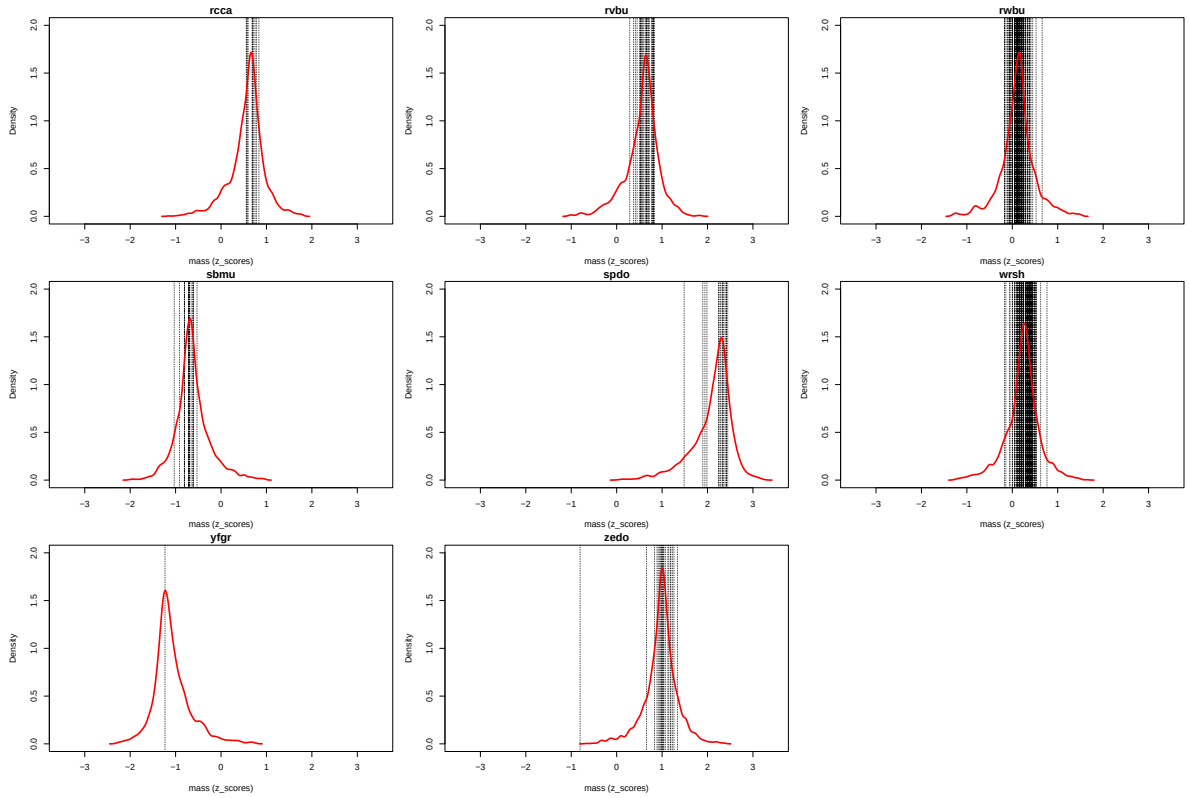


Figure S31: Posterior distributions of the body mass of species involve in seed dispersal networks in O'ahu island. Each panel belongs to one species; red density line denotes the estimated posterior distribution; vertical dotted line shows observed values that were not used to fit the model

S3.2.5 Functional space of birds

```
bird_functional_traits <-
  lapply(birds_names$species, FUN =
           function(i) {

             mass <- post_mass[, i, drop = T]
             depth <- post_depth[, i, drop = T]
             gape <- post_gape[, i, drop = T]

             labels_birds <- birds_names$Species

             indx <- which(birds_names$species == i)

             tibble(
               sp = labels_birds[indx],
               mass = mass,
               depth = depth,
               gape = gape
```

```
            )
          })

names(bird_functional_traits) <- birds_names$species

# saveRDS(bird_functional_traits, 'bird_traits.rds')
```

```
plot_bird1 <-
  ggplot() +
  geom_point(data = do.call('rbind', bird_functional_traits),
             aes(mass, depth),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = do.call('rbind', bird_functional_traits),
    geom = "raster",
    aes(mass, depth, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = bird_functional_traits$apap,
             aes(mass, depth),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = bird_functional_traits$jabw,
             aes(mass, depth),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = bird_functional_traits$rvbu,
             aes(mass, depth),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(y = 'Bill depth (z-scores)', x = 'Body mass (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))

plot_bird2 <-
  ggplot() +
  geom_point(data = do.call('rbind', bird_functional_traits),
             aes(mass, gape),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = do.call('rbind', bird_functional_traits),
    geom = "raster",
    aes(mass, gape, fill = after_stat(density)),
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = bird_functional_traits$apap,
             aes(mass, gape),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = bird_functional_traits$jabw,
             aes(mass, gape),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = bird_functional_traits$rvbu,
             aes(mass, gape),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(y = 'Bill gape (z-scores)', x = 'Body mass (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))

plot_bird3 <-
  ggplot() +
  geom_point(data = do.call('rbind', bird_functional_traits),
             aes(depth, gape),
             size = 0.1, alpha = 0.2) +
  stat_density_2d(
    data = do.call('rbind', bird_functional_traits),
    geom = "raster",
    aes(depth, gape, fill = after_stat(density)),
```

```
    contour = F, alpha = 0.7
  ) + scale_fill_viridis_c() +
  geom_point(data = bird_functional_traits$apap,
             aes(depth, gape),
             size = 0.1, color = 'tan1', alpha = 0.5) +
  geom_point(data = bird_functional_traits$jabw,
             aes(depth, gape),
             size = 0.1, color = 'lightblue', alpha = 0.5) +
  geom_point(data = bird_functional_traits$rvbu,
             aes(depth, gape),
             size = 0.1, color = 'tomato', alpha = 0.5) +
  labs(x = 'Bill depth (log)', y = 'Bill gape (z-scores)') +
  theme_classic() +
  theme(legend.position = 'none',
        text = element_text(size = 14))
```

```
plot_grid(plot_grid(plot_bird1, plot_bird2, ncol = 2),
          plot_grid(NULL, plot_bird3, NULL, ncol = 3,
                    rel_widths = c(0.5, 1.3, 0.5)),
          nrow = 2)
```

S4 Computational enviornment

```
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;  L

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] sf_1.0-21           ggplot2_3.5.2       readxl_1.4.5
 [4] cowplot_1.2.0       rethinking_2.42     posterior_1.6.1
```

```
 [7] cmdstanr_0.9.0.9000 magrittr_2.0.3      lubridate_1.9.4
[10] forcats_1.0.0       dplyr_1.1.4

loaded via a namespace (and not attached):
 [1] tensorA_0.36.2.1     generics_0.1.4       class_7.3-23
 [4] KernSmooth_2.23-26   shape_1.4.6.1        lattice_0.22-7
 [7] extrafontdb_1.0      digest_0.6.37        evaluate_1.0.4
[10] grid_4.5.1           timechange_0.3.0     RColorBrewer_1.1-3
[13] mvtnorm_1.3-3        fastmap_1.2.0        cellranger_1.1.0
[16] jsonlite_2.0.0       processx_3.8.6       e1071_1.7-16
[19] backports_1.5.0      DBI_1.2.3            ps_1.9.1
[22] viridisLite_0.4.2    scales_1.4.0         abind_1.4-8
[25] cli_3.6.5            rlang_1.1.6          units_0.8-7
[28] withr_3.0.2          yaml_2.3.10          tools_4.5.1
[31] checkmate_2.3.2      coda_0.19-4.1        vctrs_0.6.5
[34] R6_2.6.1             proxy_0.4-27         classInt_0.4-11
[37] matrixStats_1.5.0    lifecycle_1.0.4      MASS_7.3-65
[40] pkgconfig_2.0.3      pillar_1.11.0        gtable_0.3.6
[43] Rcpp_1.1.0           loo_2.8.0            glue_1.8.0
[46] xfun_0.52            tibble_3.3.0         tidyselect_1.2.1
[49] rstudioapi_0.17.1    knitr_1.50           extrafont_0.19
[52] farver_2.1.2         htmltools_0.5.8.1    rmarkdown_2.29
[55] Rttf2pt1_1.3.12      compiler_4.5.1       distributional_0.5.0
```