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

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

Appendix S2. Journal: Ecology

Table of contents

S1 Introduction

This document holds data wrangling operations for formatting plants phenology data from **Ramírez-Mejía et al. Temporal dynamics of seed-dispersal networks: Disentangling the role of direct and indirect biotic and climatic processes**. Specifically, it holds the code for fitting models to estimate richness of fruit producing species and fruit production in seven sites at the O'ahu island, Hawai'i. The document is divided in the following sections:

- Data collection
- Data wrangling operations
- Species richness model
- Fruit production model

S2 Data collection

S2.1 Fruiting phenology

Monthly fruit abundances and richness were estimated using seed rain traps placed at each sampling locality: 100 traps in MOA, WAI, and EKA; 150 traps in MTK and PAH; 148 traps in KAH; and 146 traps in TAN. Traps were placed 10 m apart, except in TAN where they were 5 m apart. Each trap had a diameter of 0.243 m and an area of 0.046 m2 and was covered with chicken wire to deter animals from removing seeds and fruits. Once a month, the cotton cloth at the bottom of the traps was collected, dried, and the seeds were identified under a stereomicroscope. To obtain local abundances, we started by dividing the number of seeds of each species in each trap by the average number of seeds per fruit (estimated from fruits collected in the field) to obtain the number of fruits per trap. We then summed the number of fruits per species across all traps at the site.

We used hierarchical Gaussian processes time series models to estimate the site-level richness of fruiting plants and fruit production at the site i during month j. To do so, we modeled the species richness (Poisson, link = log) and fruit production (Student's t, link = identity) as a function of the month and site, and included the standardized sampling effort as an offset term to account for variation on the number of seed traps used during each sampling period. For site parameters, we programmed a quadratic kernel to construct the covariance matrix reflecting the spatial correlation across sites, and used a non-centered parametrization through a Cholesky decomposition to estimate the correlated parameters. We accounted for temporal correlation and circularity of month parameters programming a periodic kernel as described above (Appendix S2: section 4.1 and 5.1). For both models, we defined prior probabilities skeptical of strong effects and executed the HMC algorithm using three Markov chains, 2000 sampling and 500 warmup iterations, and a thinning rate of 3. After verifying sampling diagnostics and the quality of model fits, we used the estimated parameters to impute missing values when we were unable to sample the month i at the site j, using as offset term the median standardized sampling effort at the site j. The estimated species richness and fruit production were used as input variables in our final model. See Appendix S2: section 4–5 for a detailed mathematical version of the models, prior probabilities employed, Stan code, and sampling diagnostic procedures.

S3 Data wrangling

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

extrafont::loadfonts(device = 'win')

source('functions_mod_diagnostics.r')
```

The following code conducts data cleaning operations to prepare both datasets for fitting the species richness and fruit production models

```r
d <- read_xlsx('SeedTraps.xlsx', col_names = T, sheet = 1)

d <- d[, c("TrapID", "Site", "Date", "Month", "MonthJef",
           "Year", "Species", '#Seeds', "#seed/#seed per fruit",
           "#fruitsTotal", "TrapTipped", "Dispersed? (Y/N)")]

d <- d[d$TrapTipped != 'Y', ]

d$`#seed/#seed per fruit`[which(d$`#seed/#seed per fruit` == 'NA')] <- 0

d$Date <- d %$% paste(day(Date), MonthJef, Year, sep = '-')
d <- d[-grep('NA', d$Date), ]

d$Date <- dmy(d$Date)

sampling_effort <-
  d[, c("TrapID", "Site", "Date")] |>
  unique() |>
  group_by(Date, Site) |>
  transmute(SampEff = length(TrapID)) |>
  unique()

sampling_effort$month <- month(sampling_effort$Date)

sampling_effort <-
  sampling_effort |>
  group_by(Site, month) |>
  transmute(SampEff = mean(SampEff)) |>
  unique()

d$Species[which(d$`Dispersed? (Y/N)` == 'N' |
                  is.na(d$`Dispersed? (Y/N)`))] <- 'NA'

d <-
  d[, c("TrapID", "Site", "Date", "Species",
      "#seed/#seed per fruit", "#fruitsTotal")] |>
  unique()

d$month <- month(d$Date)

d$date2 <- d %$% paste(year(Date), month, sep = '-')

# fruit production data

fruit_abundance <-
  d |>
  group_by(Site, date2) |>
  transmute(abun = sum(`#fruitsTotal`),
            month = month) |>
  ungroup() |>
  group_by(Site, month) |>
  transmute(abun = mean(abun)) |>
  unique()

# richness of fruit production fruits
S_fruiting <-
  d |>
  filter(Species != 'NA') |>
  select(Site, date2, Species, month) |>
  unique() |>
  group_by(Site, date2) |>
  transmute(S = length(Species),
            month = month) |>
  unique() |>
  group_by(Site, month) |>
  transmute(S = mean(S)) |>
  unique()

phenology <- full_join(S_fruiting, fruit_abundance, by = c('Site', 'month'))

phenology <- full_join(phenology, sampling_effort, by = c('Site', 'month'))

phenology$SampEff_std <- phenology$SampEff / mean(phenology$SampEff)

phenology$Site <- as.factor(phenology$Site)

dist_mat <- readRDS('mat_distance_dites.rds')

dat <- lapply(phenology, function(x) x)

dat$Site <- as.numeric(dat$Site)

dat$abun <- log(dat$abun)

dat$dist_sites <- dist_mat

dat$S <- round(dat$S)

dat$N <- length(dat$Site)
dat$N_sites <- max(dat$Site)
dat$N_months <- 12
```

S4 Species richness model

We used a times series Gaussian processes hierarchical model to estimate species richness of fruiting plants per site and month, from 2014 to 2018.

S4.1 Mathematic notation of the model

The model uses a *periodic kernel* ($COV.MAT_{ij} = \sigma_f^2 \cdot exp(-\frac{2 \cdot \sin^2(\frac{\pi|t_i - t_j|}{period})}{\zeta^2}))$ to estimate the covariance matrix of the Gaussian processes term to account temporal correlation and circularity among months. We also used a *quadratic kernel* ($COV.MAT_{ij} = \eta^2 \cdot exp(-\rho \cdot D_{ij}^2) \cdot \delta_{ij} \cdot 0.01)$ to estimate the covariance matrix of the Gaussian processes term to account spatial correlation among sites. The model includes and *offset term* consider variation of sampling effort among sites.

$$
\begin{aligned}
&\text{S } plants_i \sim Poisson(\lambda_i) \\
&log(\lambda_i) = \alpha + f_{[site_i,\ month_i]} + \theta_{site_i} + log(effort) \\
&\alpha \sim \mathcal{N}(1.5, 0.25) \\
&K_{[site,\ month]}[i,j] = \sigma^2_{[site,month]} \cdot exp(-\frac{2 \cdot \sin^2(\pi|t_i - t_j|/period)}{\zeta^2_{[site,month]}}) \\
&K_{[site,\ month]} = L_{K_{[site,\ month]}} \cdot L^T_{K_{[site,\ month]}} \\
&f_{[site,month]} = L_{K_{[site,\ month]}} \cdot \eta_{[site,month]} \\
&\zeta_{[site,month]} \sim \text{inv-gamma}(5,5) \\
&\sigma_{[site,month]} \sim cauchy(0,1) \\
&\eta_{[site,month]} \sim \mathcal{N}(1.5, 0.25) \\
&K1[i,j] = \phi^2 \cdot exp(-\rho^2 \cdot D_{ij}) + \delta_{ij} \cdot 0.01 \\
&K1 = L_{K1} \cdot L^T_{K1} \\
&\theta_{site} = K1 \cdot Z_{site} \\
&\phi \sim Exp(4) \\
&\rho \sim Exp(1) \\
&Z_{site} \sim \mathcal{N}(0,1)
\end{aligned}
$$

S4.2 Stan code

```
cat(file = 'S_phenology.stan',
    '
    functions {
  vector GP_periodic(int period,   // periodicity
                     real gamma,   // smoothing term of the GP
                     real sigma,   // noise parameter of the GP
                     vector eta){  // latent variable per month

                       int M = period;
                       matrix[M, M] K;
                       matrix[M, M] L_K;

                       for (i in 1:(M-1)) {
                         for (j in (i+1):M) {

                           real distance = abs(i - j);
                           real periodic_distance = fmin(distance, period - distance);
                           K[i, j] = sigma^2 * exp(-2*square(sin(pi()*periodic_distance/period))/gamma^2);
                           K[j, i] = K[i, j]; // filling the lower triangle
                         }
                         K[i, i] = sigma^2 + 1e-9; // small values at the diagonal for stability
                       }

                       K[M, M] = sigma^2 + 1e-9; // small values at the begining and end of the matrix
                       return(cholesky_decompose(K) * eta);
                     }

  matrix GP_quadratic(matrix x,
                      real eta,
                      real rho,
                      real delta) {

                        int N = dims(x)[1];
                        matrix[N, N] K;
                        matrix[N, N] L_K;

                        for (i in 1:(N-1)) {
                          K[i, i] = eta + delta; // small values to the diagonal
                          for (j in (i+1):N) {
                            K[i, j] = eta * exp(-rho * square(x[i, j]));
                            K[j, i] = K[i, j]; // filling low triangle
                          }
                        }

                      K[N, N] = eta + delta;
                      L_K = cholesky_decompose(K);
                      return L_K;
                      }
}

data {
  int N;
  int N_sites;
  int N_months;
  array[N] int Site;
  array[N] int month;
  array[N] int S;
  vector[N] abun;
  vector[N] SampEff_std;
  matrix[N_sites, N_sites] dist_sites;
}

parameters {

  real alpha;
  //real<lower = 0> sigma;
  // periodic GP
  vector<lower = 0>[N_sites] gamma_f;
  vector<lower = 0>[N_sites] sigma_f;
  matrix[N_months, N_sites] eta_f;

  // Quadratic GP
  vector[N_sites] z_sites;
  real<lower = 0> eta;
  real<lower = 0> rho;

}

transformed parameters {
  // periodic GP
  matrix[N_months, N_sites] f;

  for (i in 1:N_sites) {
    f[, i] = GP_periodic(
      12,
      gamma_f[i],
      sigma_f[i],
      eta_f[,i]
    );
  }

  // Quadratic GP

  vector[N_sites] theta;
  matrix[N_sites, N_sites] L_K_theta;
  L_K_theta = GP_quadratic(dist_sites,
                           eta,
                           rho,
                           0.001);
  theta = L_K_theta * z_sites;
}

model {
  alpha ~ normal(1.5, 0.25);
  //sigma ~ exponential(1);

  to_vector(eta_f) ~ normal(1.5, 0.25);
  gamma_f ~ inv_gamma(5, 5);
  sigma_f ~ cauchy(0, 1);

  z_sites ~ normal(0, 1);
  eta ~ exponential(4);
  rho ~ exponential(1);

  // likelihood

  for (i in 1:N) {
   S[i] ~ poisson(exp(alpha + f[month[i], Site[i]] +
                           theta[Site[i]] + log(SampEff_std[i])));
  }
}

generated quantities {
  vector[N] mu;
  array[N] int ppcheck;

  for (i in 1:N) {
    mu[i] = exp(alpha + f[month[i], Site[i]] + theta[Site[i]]) .*SampEff_std[i];
  }
  ppcheck = poisson_rng(mu);

}
    ')
```

S4.3 Fitting the model

```r
file <- paste0(getwd(), '/S_phenology.stan')

fit_S <- cmdstan_model(file, compile = T)
```

```
Warning in readLines(stan_file): incomplete final line found on
'/Users/andres/Documents/github_repos/HAWAII_temporal_dynamics_NETWORK_LEVEL/S_phenology.sta
```

```r
mod_S <-
  fit_S$sample(
    data = dat,
    iter_sampling = 2e3,
    iter_warmup = 500,
    chains = 3,
    parallel_chains = 3,
    thin = 3,
    seed = 5
  )
```

```
Running MCMC with 3 parallel chains...

Chain 1 Iteration:    1 / 2500 [  0%]  (Warmup)

Chain 1 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 1 Exception: Exception: cholesky_decompose: A is not symmetric. A[1,2] = nan, but A[2,

Chain 1 If this warning occurs sporadically, such as for highly constrained variable types li

Chain 1 but if this warning occurs often then your model may be either severely ill-conditio

Chain 1

Chain 1 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 1 Exception: Exception: cholesky_decompose: A is not symmetric. A[1,2] = nan, but A[2,

Chain 1 If this warning occurs sporadically, such as for highly constrained variable types li
Chain 1 but if this warning occurs often then your model may be either severely ill-conditio

Chain 1

Chain 1 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 1 Exception: Exception: cholesky_decompose: A is not symmetric. A[1,2] = nan, but A[2,

Chain 1 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 1 but if this warning occurs often then your model may be either severely ill-conditio

Chain 1

Chain 1 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 1 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 1 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 1 but if this warning occurs often then your model may be either severely ill-conditio

Chain 1

Chain 1 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 1 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 1 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 1 but if this warning occurs often then your model may be either severely ill-conditio

Chain 1

Chain 2 Iteration:    1 / 2500 [  0%]  (Warmup)

Chain 2 Informational Message: The current Metropolis proposal is about to be rejected becau
Chain 2 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 2 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 2 but if this warning occurs often then your model may be either severely ill-conditio

Chain 2

Chain 2 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 2 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 2 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 2 but if this warning occurs often then your model may be either severely ill-conditio

Chain 2

Chain 3 Iteration:    1 / 2500 [  0%]  (Warmup)

Chain 3 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 3 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 3 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 3 but if this warning occurs often then your model may be either severely ill-conditio

Chain 3

Chain 3 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 3 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 3 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 3 but if this warning occurs often then your model may be either severely ill-conditio
Chain 3

Chain 3 Informational Message: The current Metropolis proposal is about to be rejected becaus

Chain 3 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 3 If this warning occurs sporadically, such as for highly constrained variable types li

Chain 3 but if this warning occurs often then your model may be either severely ill-conditio

Chain 3

Chain 1 Iteration:  100 / 2500 [  4%]  (Warmup)
Chain 3 Iteration:  100 / 2500 [  4%]  (Warmup)
Chain 2 Iteration:  100 / 2500 [  4%]  (Warmup)
Chain 1 Iteration:  200 / 2500 [  8%]  (Warmup)
Chain 2 Iteration:  200 / 2500 [  8%]  (Warmup)
Chain 3 Iteration:  200 / 2500 [  8%]  (Warmup)
Chain 1 Iteration:  300 / 2500 [ 12%]  (Warmup)
Chain 2 Iteration:  300 / 2500 [ 12%]  (Warmup)
Chain 3 Iteration:  300 / 2500 [ 12%]  (Warmup)
Chain 1 Iteration:  400 / 2500 [ 16%]  (Warmup)
Chain 2 Iteration:  400 / 2500 [ 16%]  (Warmup)
Chain 3 Iteration:  400 / 2500 [ 16%]  (Warmup)
Chain 1 Iteration:  500 / 2500 [ 20%]  (Warmup)
Chain 1 Iteration:  501 / 2500 [ 20%]  (Sampling)
Chain 2 Iteration:  500 / 2500 [ 20%]  (Warmup)
Chain 2 Iteration:  501 / 2500 [ 20%]  (Sampling)
Chain 1 Iteration:  600 / 2500 [ 24%]  (Sampling)
Chain 3 Iteration:  500 / 2500 [ 20%]  (Warmup)
Chain 3 Iteration:  501 / 2500 [ 20%]  (Sampling)
Chain 1 Iteration:  700 / 2500 [ 28%]  (Sampling)
Chain 2 Iteration:  600 / 2500 [ 24%]  (Sampling)
Chain 1 Iteration:  800 / 2500 [ 32%]  (Sampling)
Chain 3 Iteration:  600 / 2500 [ 24%]  (Sampling)
Chain 1 Iteration:  900 / 2500 [ 36%]  (Sampling)
Chain 2 Iteration:  700 / 2500 [ 28%]  (Sampling)
Chain 1 Iteration: 1000 / 2500 [ 40%]  (Sampling)
Chain 3 Iteration:  700 / 2500 [ 28%]  (Sampling)
Chain 1 Iteration: 1100 / 2500 [ 44%]  (Sampling)
Chain 2 Iteration:  800 / 2500 [ 32%]  (Sampling)
Chain 3 Iteration:  800 / 2500 [ 32%]  (Sampling)
Chain 1 Iteration: 1200 / 2500 [ 48%]  (Sampling)
Chain 1 Iteration: 1300 / 2500 [ 52%]  (Sampling)
Chain 2 Iteration:  900 / 2500 [ 36%]  (Sampling)
Chain 3 Iteration:  900 / 2500 [ 36%]  (Sampling)
Chain 1 Iteration: 1400 / 2500 [ 56%]  (Sampling)
Chain 1 Iteration: 1500 / 2500 [ 60%]  (Sampling)
Chain 2 Iteration: 1000 / 2500 [ 40%]  (Sampling)
Chain 3 Iteration: 1000 / 2500 [ 40%]  (Sampling)
Chain 1 Iteration: 1600 / 2500 [ 64%]  (Sampling)
Chain 1 Iteration: 1700 / 2500 [ 68%]  (Sampling)
Chain 2 Iteration: 1100 / 2500 [ 44%]  (Sampling)
Chain 3 Iteration: 1100 / 2500 [ 44%]  (Sampling)
Chain 1 Iteration: 1800 / 2500 [ 72%]  (Sampling)
Chain 1 Iteration: 1900 / 2500 [ 76%]  (Sampling)
Chain 2 Iteration: 1200 / 2500 [ 48%]  (Sampling)
Chain 3 Iteration: 1200 / 2500 [ 48%]  (Sampling)
Chain 1 Iteration: 2000 / 2500 [ 80%]  (Sampling)
Chain 2 Iteration: 1300 / 2500 [ 52%]  (Sampling)
Chain 1 Iteration: 2100 / 2500 [ 84%]  (Sampling)
Chain 3 Iteration: 1300 / 2500 [ 52%]  (Sampling)
Chain 1 Iteration: 2200 / 2500 [ 88%]  (Sampling)
Chain 2 Iteration: 1400 / 2500 [ 56%]  (Sampling)
Chain 1 Iteration: 2300 / 2500 [ 92%]  (Sampling)
Chain 3 Iteration: 1400 / 2500 [ 56%]  (Sampling)
Chain 1 Iteration: 2400 / 2500 [ 96%]  (Sampling)
Chain 2 Iteration: 1500 / 2500 [ 60%]  (Sampling)
Chain 1 Iteration: 2500 / 2500 [100%]  (Sampling)
Chain 3 Iteration: 1500 / 2500 [ 60%]  (Sampling)
Chain 1 finished in 7.2 seconds.
Chain 2 Iteration: 1600 / 2500 [ 64%]  (Sampling)
Chain 3 Iteration: 1600 / 2500 [ 64%]  (Sampling)
Chain 2 Iteration: 1700 / 2500 [ 68%]  (Sampling)
Chain 3 Iteration: 1700 / 2500 [ 68%]  (Sampling)
Chain 2 Iteration: 1800 / 2500 [ 72%]  (Sampling)
Chain 3 Iteration: 1800 / 2500 [ 72%]  (Sampling)
Chain 2 Iteration: 1900 / 2500 [ 76%]  (Sampling)
Chain 3 Iteration: 1900 / 2500 [ 76%]  (Sampling)
Chain 2 Iteration: 2000 / 2500 [ 80%]  (Sampling)
Chain 3 Iteration: 2000 / 2500 [ 80%]  (Sampling)
Chain 2 Iteration: 2100 / 2500 [ 84%]  (Sampling)
Chain 3 Iteration: 2100 / 2500 [ 84%]  (Sampling)
Chain 2 Iteration: 2200 / 2500 [ 88%]  (Sampling)
Chain 3 Iteration: 2200 / 2500 [ 88%]  (Sampling)
Chain 2 Iteration: 2300 / 2500 [ 92%]  (Sampling)
Chain 3 Iteration: 2300 / 2500 [ 92%]  (Sampling)
Chain 2 Iteration: 2400 / 2500 [ 96%]  (Sampling)
Chain 3 Iteration: 2400 / 2500 [ 96%]  (Sampling)
Chain 2 Iteration: 2500 / 2500 [100%]  (Sampling)
Chain 3 Iteration: 2500 / 2500 [100%]  (Sampling)
Chain 2 finished in 11.8 seconds.
Chain 3 finished in 11.8 seconds.

All 3 chains finished successfully.
Mean chain execution time: 10.3 seconds.
Total execution time: 12.0 seconds.


Warning: 3 of 2001 (0.0%) transitions ended with a divergence.
See https://mc-stan.org/misc/warnings for details.
```

S4.4 Sampling diagnostics

```r
summary_S <- mod_S$summary()
mod_diagnostics(mod_S, summary_S)
```

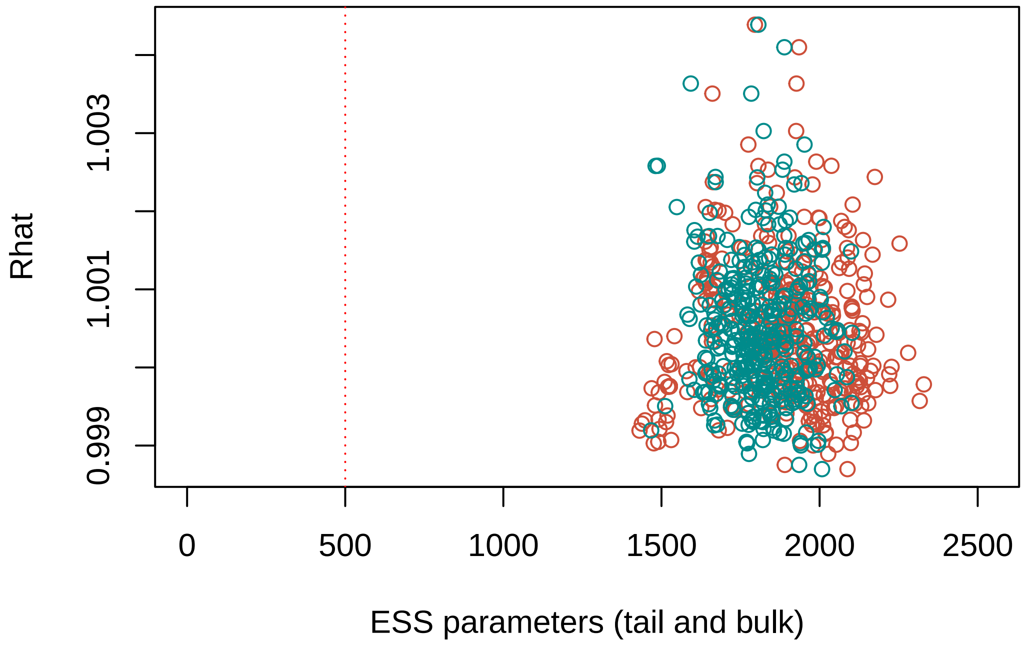

Figure S1: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_S <- mod_S$draws('ppcheck', format = 'matrix')
plot(density(dat$S), ylim = c(0, 0.20), xlim = c(-2, 20),
     xlab = 'S (fruiting plants)', main = '')
for (i in 1:200) lines(density(ppcheck_S[i, ]), lwd = 0.1)
lines(density(dat$S), col = 'red', lwd = 2)
```

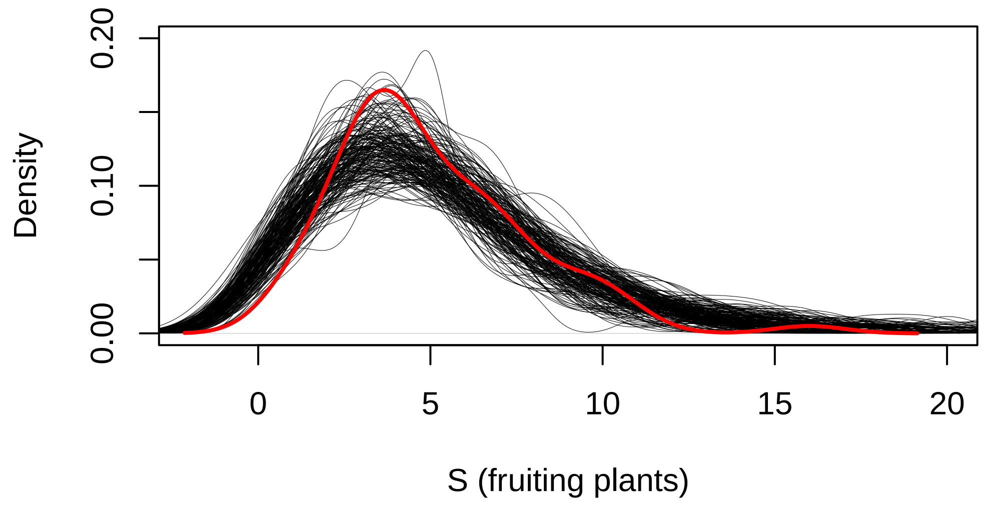

Figure S2: Posterior predictive checks of the model describing monthly average plant richness producing fruits in seven sites in the Oahu island.

S4.5 Extracting posterior distribution

This code extract the posterior distributions and use the model's structure to estimate monthly species richness per site. It shows observed and imputed values.

```r
mu_S <- mod_S$draws('mu', format = 'df')

mu_S <- mu_S[, grep('mu', colnames(mu_S))]
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```r
mu_S <- lapply(mu_S, FUN =
                 function(x) {
                   tibble(mu = mean(x),
                          li = quantile(x, 0.025),
                          ls = quantile(x, 0.975))
                 })

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

colnames(mu_S) <- paste0('S_', colnames(mu_S))

phenology <- cbind(phenology, mu_S)
```

```r
phenology$date <- dmy(paste0('01-', phenology$month, '-2025'))

inter_data <- readRDS('data_for_models.rds')
inter_data <- unique(inter_data[, c("site2", "month")])
inter_data$cod <- inter_data %$% paste0(site2, '_', month)

phenology$cod <- phenology %$% paste0(Site, '_', month)

impute_dates <- !(inter_data$cod %in% phenology$cod)

inter_data$site2 <- as.factor(inter_data$site2)
inter_data$site <- as.numeric(inter_data$site2)

post_S <- mod_S$draws(c('alpha', 'f', 'theta'), format = 'df')
post_S <- lapply(c('alpha', 'f', 'theta'), FUN =
                   function(x) {
                     post_S[, grep(x, colnames(post_S))]
                   })
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
Warning: Dropping 'draws_df' class as required metadata was removed.
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```r
names(post_S) <- c('alpha', 'f', 'theta')

pred_S <-
  lapply(1:7, FUN =
           function(i) {

             indx <- which(i == dat$Site)[1]
             S <- phenology$Site[indx]

             site <-
               lapply(1:12, FUN =
                        function(j) {

                          GP <- paste0('f[', j, ',', i, ']')
                          t <- paste0('theta[', i, ']')

                          cod <- paste0(S, '_', j)
                          indx_SEff <- which(cod == phenology$cod)
                          indx_months <- which(i == phenology$month)

                          est <-
                            with(post_S,
                                 {
                                   alpha[, 1, drop = T] +
                                     f[, GP, drop = T] +
                                     theta[, t, drop = T]
                                 })

                          if (length(indx_SEff) > 0) {
                            est <- exp(est) * dat$SampEff_std[indx_SEff]
                          } else {
                            est <- exp(est) * median(dat$SampEff_std[indx_months])
                          }

                          est <-
                            tibble(Site = i,
                                   date = dmy(paste0('01-', j, '-2025')),
                                   S_mu = sample(est, 1e3))
                          est
                        })

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

             site$Site <- S
             site
           })

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

pred_S$z_s <- (pred_S$S_mu - mean(dat$S)) / sd(dat$S)

est_S <-
  lapply(1:7, FUN =
         function(i) {

           indx <- which(i == dat$Site)[1]
           S <- phenology$Site[indx]

           site <-
             lapply(1:12, FUN =
                      function(j) {

                        GP <- paste0('f[', j, ',', i, ']')
                        t <- paste0('theta[', i, ']')

                        cod <- paste0(S, '_', j)
                        indx_SEff <- which(cod == phenology$cod)
                        indx_months <- which(i == phenology$month)

                        est <-
                          with(post_S,
                               {
                                 alpha[, 1, drop = T] +
                                   f[, GP, drop = T] +
                                   theta[, t, drop = T]
                               })

                        if (length(indx_SEff) > 0) {
                          est <- exp(est) * dat$SampEff_std[indx_SEff]
                        } else {
                          est <- exp(est) * median(dat$SampEff_std[indx_months])
                        }

                        est <-
                          tibble(Site = i,
                                 date = dmy(paste0('01-', j, '-2025')),
                                 S_mu = mean(est),
                                 S_li = quantile(est, 0.025),
                                 S_ls = quantile(est, 0.975))
                        est
                      })

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

           site$Site <- S
           site
         })

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

S4.6 Plotting posterior distribution

```
ggplot() +
  geom_line(data = phenology, aes(date, S_mu, color = 'No imputation')) +
  geom_point(data = phenology, aes(date, S_mu, color = 'No imputation'),
             size = 1.5) +
  geom_ribbon(data = phenology,
              aes(date, S_mu, ymin = S_li, ymax = S_ls,
                  fill = 'No imputation'),
              alpha = 0.3) +
  geom_line(data = est_S, aes(date, S_mu, color = 'Imputation'), lwd = 0.1) +
  geom_point(data = est_S, aes(date, S_mu, color = 'Imputation'), size = 0.2) +
  geom_ribbon(data = est_S,
              aes(date, S_mu, ymin = S_li, ymax = S_ls, fill = 'Imputation'),
              alpha = 0.3) +
  scale_x_date(date_labels = '%b', date_breaks = '2 month') +
  facet_wrap(~Site) +
  labs(y = 'S (fruiting plants)') +
  theme_bw()
```

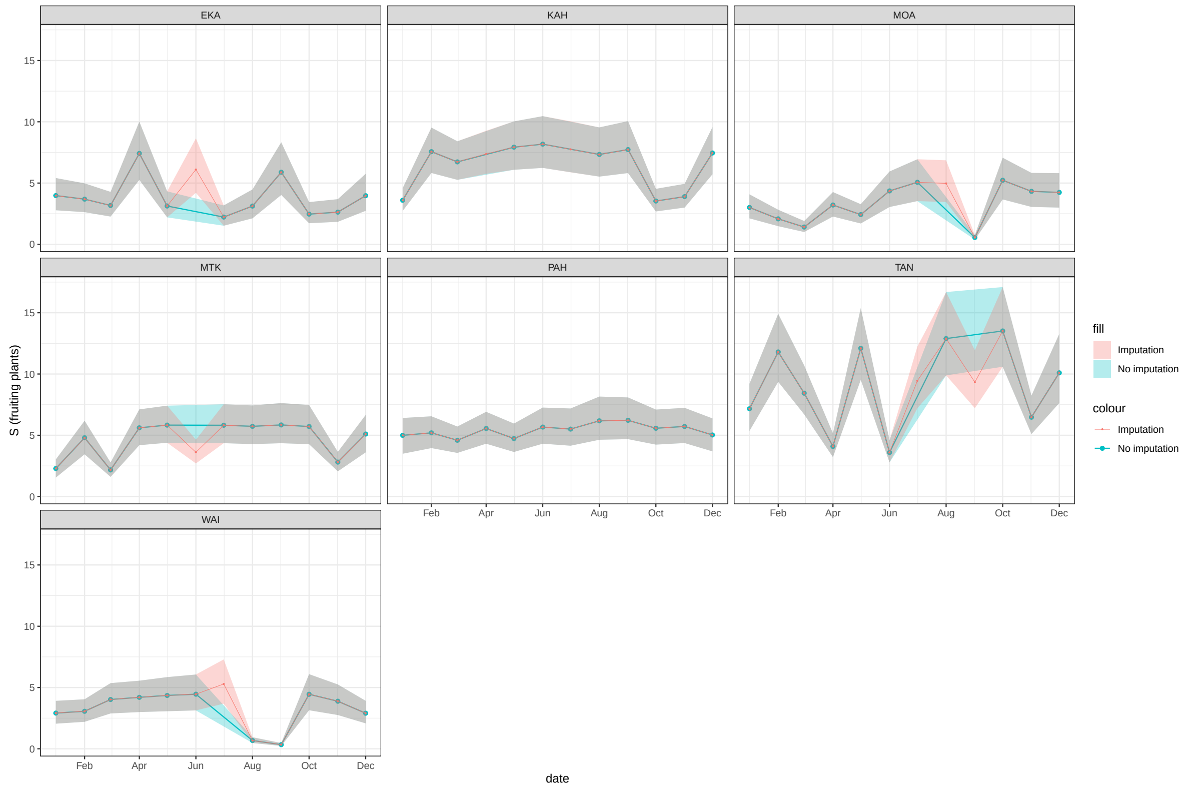

Figure S3: Monthly species richnes of fruiting plants in seven sites at the Oahu island. Non-overlapped red dotes indicates imputed months

S5 Fruit production

We used a times series Gaussian processes hierarchical model to estimate average fruit production per site and month, from 2014 to 2018.

S5.1 Mathematic notation of the model

The model uses a *periodic kernel* ($COV.MAT_{ij} = \sigma_f^2 \cdot exp(-\frac{2 \cdot \sin^2(\frac{\pi|t_i - t_j|}{period})}{\zeta^2})$) to estimate the covariance matrix of the Gaussian processes term to account temporal correlation and circularity among months. We also used a *quadratic kernel* ($COV.MAT_{ij} = \eta^2 \cdot exp(-\rho \cdot D_{ij}^2) \cdot \delta_{ij} \cdot 0.01$) to estimate the covariance matrix of the Gaussian processes term to account spatial correlation among sites. The model includes and *offset term* consider variation of sampling effort among sites.

$$
\begin{aligned}
&\text{fruit production}_i \sim \text{Student-T}(\nu = 6, \mu_i, \sigma) \\
&\mu_i = \alpha + f_{[site_i, month_i]} + \theta_{site} + \text{effort} \\
&\alpha \sim \mathcal{N}(0, 1) \\
&K_{[site,month]}[i, j] = \sigma^2_{[site,month]} \cdot exp(-\frac{2 \cdot \sin^2(\pi|t_i - t_j|/period)}{\zeta^2_{[site,month]}}) \\
&K_{[site,month]} = L_{K_{[site,month]}} \cdot L^T_{K_{[site,month]}} \\
&f_{[site,month]} = L_{K_{[site,month]}} \cdot \eta_{[site,month]} \\
&\zeta_{[site,month]} \sim \text{inv-gamma}(5, 5) \\
&\sigma_{[sute,month]} \sim cauchy(0, 1) \\
&\eta_{[site,month]} \sim \mathcal{N}(0, 1) \\
&K1_{[i,j]} = \phi^2 \cdot exp(-\rho^2 \cdot D_{ij}) + \delta_{i,j} \cdot 0.01 \\
&K1 = L_{K1} \cdot L^T_{K1} \\
&\theta_{site} = L_{K1} \cdot Z_{site} \\
&\phi \sim exp(4) \\
&\rho \sim exp(1) \\
&Z_{site} \sim \mathcal{N}(0, 1)
\end{aligned}
$$

S5.2 Stan code

```
cat(file = 'abun_phenology.stan',
    '
    functions {
  vector GP_periodic(int period,   // periodicity
                     real gamma,   // smoothing term of the GP
                     real sigma,   // noise parameter of the GP
                     vector eta){  // latent variable per month

                       int M = period;
                       matrix[M, M] K;
                       matrix[M, M] L_K;

                       for (i in 1:(M-1)) {
                         for (j in (i+1):M) {

                           real distance = abs(i - j);
                           real periodic_distance = fmin(distance, period - distance);
                           K[i, j] = sigma^2 * exp(-2*square(sin(pi()*periodic_distance/period))/gamma^2);
                           K[j, i] = K[i, j]; // filling the lower triangle
                         }
                         K[i, i] = sigma^2 + 1e-9; // small values at the diagonal for stability
                       }

                       K[M, M] = sigma^2 + 1e-9; // small values at the begining and end of the matrix
                       return(cholesky_decompose(K) * eta);
                     }

  matrix GP_quadratic(matrix x,
                      real eta,
                      real rho,
                      real delta) {

                        int N = dims(x)[1];
                        matrix[N, N] K;
                        matrix[N, N] L_K;

                        for (i in 1:(N-1)) {
                          K[i, i] = eta + delta; // small values to the diagonal
                          for (j in (i+1):N) {
                            K[i, j] = eta * exp(-rho * square(x[i, j]));
                            K[j, i] = K[i, j]; // filling low triangle
                          }
                        }

                      K[N, N] = eta + delta;
                      L_K = cholesky_decompose(K);
                      return L_K;
                      }
}

data {
  int N;
  int N_sites;
  int N_months;
  array[N] int Site;
  array[N] int month;
  array[N] int S;
  vector[N] abun;
  vector[N] SampEff_std;
  matrix[N_sites, N_sites] dist_sites;
}

parameters {

  real alpha;
  real<lower = 0> sigma;
  // periodic GP
  vector<lower = 0>[N_sites] gamma_f;
  vector<lower = 0>[N_sites] sigma_f;
  matrix[N_months, N_sites] eta_f;

  // Quadratic GP
  vector[N_sites] z_sites;
  real<lower = 0> eta;
  real<lower = 0> rho;

}

transformed parameters {
  // periodic GP
  matrix[N_months, N_sites] f;

  for (i in 1:N_sites) {
    f[, i] = GP_periodic(
      12,
      gamma_f[i],
      sigma_f[i],
      eta_f[,i]
    );
  }

  // Quadratic GP

  vector[N_sites] theta;
  matrix[N_sites, N_sites] L_K_theta;
  L_K_theta = GP_quadratic(dist_sites,
                           eta,
                           rho,
                           0.001);
  theta = L_K_theta * z_sites;
}

model {
  alpha ~ normal(0, 1);
  sigma ~ exponential(1);

  to_vector(eta_f) ~ normal(0, 0.5);
  gamma_f ~ inv_gamma(5, 5);
  sigma_f ~ cauchy(0, 1);

  z_sites ~ normal(0, 1);
  eta ~ exponential(4);
  rho ~ exponential(1);

  // likelihood

  for (i in 1:N) {
   abun[i] ~ student_t(6, alpha + f[month[i], Site[i]] +
                          theta[Site[i]] + SampEff_std[i],
                       sigma);
  }
}

generated quantities {
  vector[N] mu;
  array[N] real ppcheck;

  for (i in 1:N) {
    mu[i] = alpha + f[month[i], Site[i]] +
            theta[Site[i]] + SampEff_std[i];
  }
  ppcheck = student_t_rng(7, mu, sigma);
```

```
}
    ')
```

S5.3 Fitting the model

```r
dat$abun <- as.vector(scale(dat$abun))

file <- paste0(getwd(), '/abun_phenology.stan')

fit_ABU <- cmdstan_model(file, compile = T)
```

```
Warning in readLines(stan_file): incomplete final line found on
'/Users/andres/Documents/github_repos/HAWAII_temporal_dynamics_NETWORK_LEVEL/abun_phenology.s
```

```r
mod_ABU <-
  fit_ABU$sample(
    data = dat,
    iter_sampling = 2e3,
    iter_warmup = 500,
    chains = 3,
    parallel_chains = 3,
    thin = 3,
    seed = 5
  )
```

```
Running MCMC with 3 parallel chains...

Chain 1 Iteration:    1 / 2500 [  0%]  (Warmup)

Chain 1 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 1 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 1 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 1 but if this warning occurs often then your model may be either severely ill-conditio

Chain 1

Chain 2 Iteration:    1 / 2500 [  0%]  (Warmup)

Chain 2 Informational Message: The current Metropolis proposal is about to be rejected becau
Chain 2 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 2 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 2 but if this warning occurs often then your model may be either severely ill-conditio

Chain 2

Chain 2 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 2 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 2 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 2 but if this warning occurs often then your model may be either severely ill-conditio

Chain 2

Chain 3 Iteration:    1 / 2500 [  0%]  (Warmup)

Chain 3 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 3 Exception: Exception: cholesky_decompose: Matrix m is not positive definite (in '/va

Chain 3 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 3 but if this warning occurs often then your model may be either severely ill-conditio

Chain 3
Chain 1 Iteration:  100 / 2500 [  4%]  (Warmup)
Chain 2 Iteration:  100 / 2500 [  4%]  (Warmup)
Chain 3 Iteration:  100 / 2500 [  4%]  (Warmup)
Chain 1 Iteration:  200 / 2500 [  8%]  (Warmup)
Chain 2 Iteration:  200 / 2500 [  8%]  (Warmup)
Chain 3 Iteration:  200 / 2500 [  8%]  (Warmup)
Chain 1 Iteration:  300 / 2500 [ 12%]  (Warmup)
Chain 2 Iteration:  300 / 2500 [ 12%]  (Warmup)
Chain 3 Iteration:  300 / 2500 [ 12%]  (Warmup)
Chain 1 Iteration:  400 / 2500 [ 16%]  (Warmup)
Chain 2 Iteration:  400 / 2500 [ 16%]  (Warmup)
Chain 3 Iteration:  400 / 2500 [ 16%]  (Warmup)
Chain 1 Iteration:  500 / 2500 [ 20%]  (Warmup)
Chain 1 Iteration:  501 / 2500 [ 20%]  (Sampling)
Chain 2 Iteration:  500 / 2500 [ 20%]  (Warmup)
Chain 2 Iteration:  501 / 2500 [ 20%]  (Sampling)
Chain 3 Iteration:  500 / 2500 [ 20%]  (Warmup)
Chain 3 Iteration:  501 / 2500 [ 20%]  (Sampling)
Chain 1 Iteration:  600 / 2500 [ 24%]  (Sampling)
Chain 2 Iteration:  600 / 2500 [ 24%]  (Sampling)
Chain 3 Iteration:  600 / 2500 [ 24%]  (Sampling)
Chain 2 Iteration:  700 / 2500 [ 28%]  (Sampling)
Chain 1 Iteration:  700 / 2500 [ 28%]  (Sampling)
Chain 2 Iteration:  800 / 2500 [ 32%]  (Sampling)
Chain 3 Iteration:  700 / 2500 [ 28%]  (Sampling)
Chain 2 Iteration:  900 / 2500 [ 36%]  (Sampling)
Chain 1 Iteration:  800 / 2500 [ 32%]  (Sampling)
Chain 3 Iteration:  800 / 2500 [ 32%]  (Sampling)
Chain 2 Iteration: 1000 / 2500 [ 40%]  (Sampling)
Chain 1 Iteration:  900 / 2500 [ 36%]  (Sampling)
Chain 2 Iteration: 1100 / 2500 [ 44%]  (Sampling)
Chain 3 Iteration:  900 / 2500 [ 36%]  (Sampling)
Chain 1 Iteration: 1000 / 2500 [ 40%]  (Sampling)
Chain 2 Iteration: 1200 / 2500 [ 48%]  (Sampling)
Chain 3 Iteration: 1000 / 2500 [ 40%]  (Sampling)
Chain 2 Iteration: 1300 / 2500 [ 52%]  (Sampling)
Chain 1 Iteration: 1100 / 2500 [ 44%]  (Sampling)
Chain 2 Iteration: 1400 / 2500 [ 56%]  (Sampling)
Chain 3 Iteration: 1100 / 2500 [ 44%]  (Sampling)
Chain 1 Iteration: 1200 / 2500 [ 48%]  (Sampling)
Chain 2 Iteration: 1500 / 2500 [ 60%]  (Sampling)
Chain 3 Iteration: 1200 / 2500 [ 48%]  (Sampling)
Chain 2 Iteration: 1600 / 2500 [ 64%]  (Sampling)
Chain 1 Iteration: 1300 / 2500 [ 52%]  (Sampling)
Chain 2 Iteration: 1700 / 2500 [ 68%]  (Sampling)
Chain 3 Iteration: 1300 / 2500 [ 52%]  (Sampling)
Chain 1 Iteration: 1400 / 2500 [ 56%]  (Sampling)
Chain 2 Iteration: 1800 / 2500 [ 72%]  (Sampling)
Chain 3 Iteration: 1400 / 2500 [ 56%]  (Sampling)
Chain 2 Iteration: 1900 / 2500 [ 76%]  (Sampling)
Chain 1 Iteration: 1500 / 2500 [ 60%]  (Sampling)
Chain 3 Iteration: 1500 / 2500 [ 60%]  (Sampling)
Chain 2 Iteration: 2000 / 2500 [ 80%]  (Sampling)
Chain 1 Iteration: 1600 / 2500 [ 64%]  (Sampling)
Chain 2 Iteration: 2100 / 2500 [ 84%]  (Sampling)
Chain 3 Iteration: 1600 / 2500 [ 64%]  (Sampling)
Chain 2 Iteration: 2200 / 2500 [ 88%]  (Sampling)
Chain 1 Iteration: 1700 / 2500 [ 68%]  (Sampling)
Chain 3 Iteration: 1700 / 2500 [ 68%]  (Sampling)
Chain 2 Iteration: 2300 / 2500 [ 92%]  (Sampling)
Chain 1 Iteration: 1800 / 2500 [ 72%]  (Sampling)
Chain 2 Iteration: 2400 / 2500 [ 96%]  (Sampling)
Chain 3 Iteration: 1800 / 2500 [ 72%]  (Sampling)
Chain 1 Iteration: 1900 / 2500 [ 76%]  (Sampling)
Chain 2 Iteration: 2500 / 2500 [100%]  (Sampling)
Chain 2 finished in 9.8 seconds.
Chain 3 Iteration: 1900 / 2500 [ 76%]  (Sampling)
Chain 1 Iteration: 2000 / 2500 [ 80%]  (Sampling)
Chain 3 Iteration: 2000 / 2500 [ 80%]  (Sampling)
Chain 1 Iteration: 2100 / 2500 [ 84%]  (Sampling)
Chain 3 Iteration: 2100 / 2500 [ 84%]  (Sampling)
Chain 1 Iteration: 2200 / 2500 [ 88%]  (Sampling)
Chain 3 Iteration: 2200 / 2500 [ 88%]  (Sampling)
Chain 1 Iteration: 2300 / 2500 [ 92%]  (Sampling)
Chain 3 Iteration: 2300 / 2500 [ 92%]  (Sampling)
Chain 1 Iteration: 2400 / 2500 [ 96%]  (Sampling)
Chain 3 Iteration: 2400 / 2500 [ 96%]  (Sampling)
Chain 1 Iteration: 2500 / 2500 [100%]  (Sampling)
Chain 1 finished in 12.9 seconds.
Chain 3 Iteration: 2500 / 2500 [100%]  (Sampling)
Chain 3 finished in 13.0 seconds.

All 3 chains finished successfully.
Mean chain execution time: 11.9 seconds.
Total execution time: 13.1 seconds.
```

S5.4 Sampling diagnostics

```
summary_ABU <- mod_ABU$summary()
mod_diagnostics(mod_ABU, summary_ABU)
```

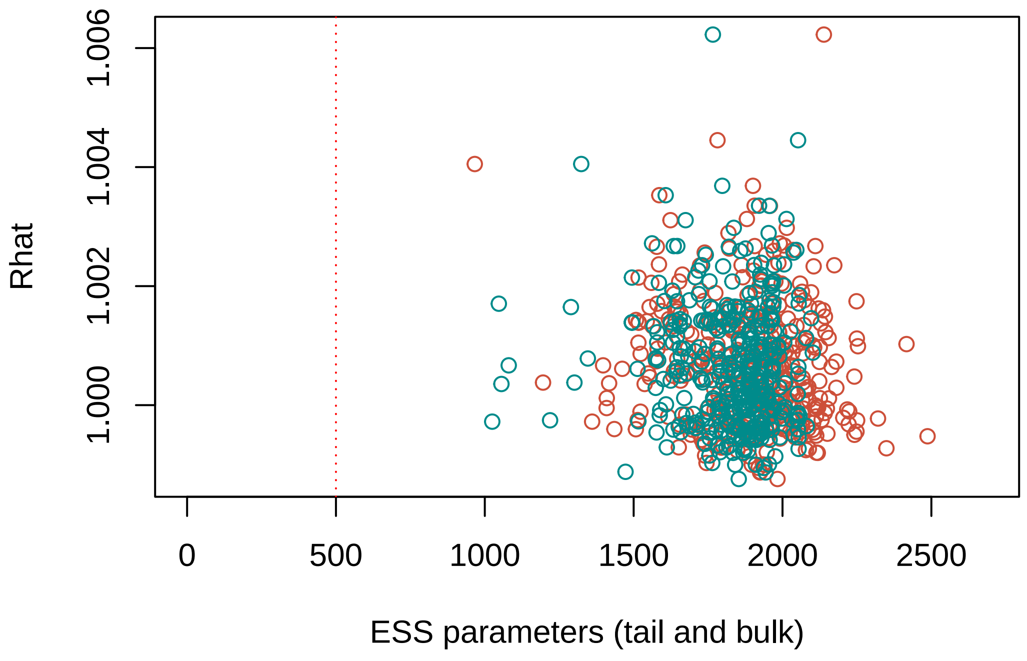

Figure S4: Rhat values vs effective sampling size (ess). Rhat < 1.05 indicates that the Markov Chains converged to the same stationary distribution. ess must be at least 100 per chain in order to be reliable

```
ppcheck_ABU <- mod_ABU$draws('ppcheck', format = 'matrix')
plot(density(dat$abun), ylim = c(0, 0.6), xlim = c(-5, 5),
     xlab = 'Fruit abundance', main = '')
for (i in 1:200) lines(density(ppcheck_ABU[i, ]), lwd = 0.1)
lines(density(dat$abun), col = 'red', lwd = 2)
```

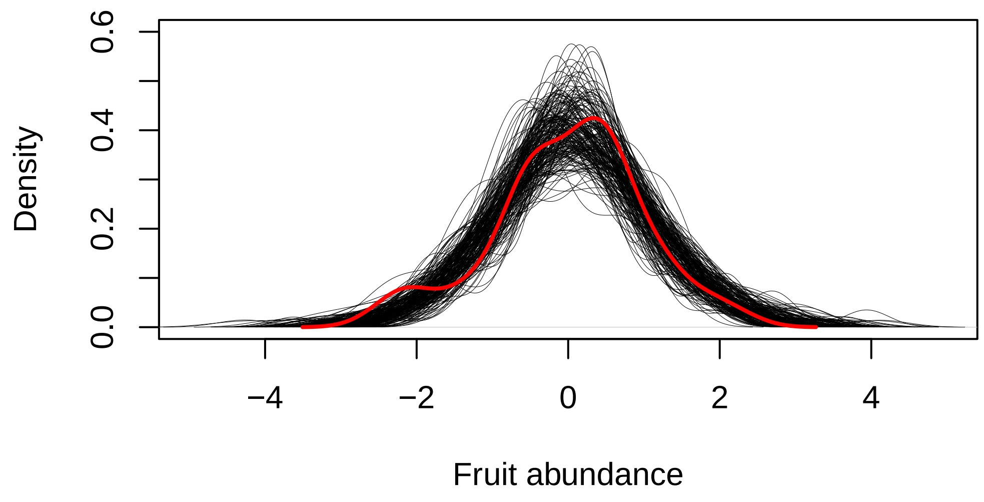

Figure S5: Posterior predictive checks of the model describing monthly average fruit production in seven sites in the Oahu island.

S5.5 Extracting posterior distribution

This code extract the posterior distributions and use the model's structure to estimate monthly fruit production per site. It shows observed and imputed values.

```
mu_ABU <- mod_ABU$draws('mu', format = 'df')

mu_ABU <- mu_ABU[, grep('mu', colnames(mu_ABU))]
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
mu_ABU <- lapply(mu_ABU, FUN =
                 function(x) {
                   tibble(mu = mean(x),
                          li = quantile(x, 0.025),
                          ls = quantile(x, 0.975))
                 })

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

colnames(mu_ABU) <- paste0('ABU_', colnames(mu_ABU))

phenology <- cbind(phenology, mu_ABU)
```

```
post_ABU <- mod_ABU$draws(c('alpha', 'f', 'theta'), format = 'df')
post_ABU <- lapply(c('alpha', 'f', 'theta'), FUN =
                   function(x) {
                     post_ABU[, grep(x, colnames(post_ABU))]
                   })
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
Warning: Dropping 'draws_df' class as required metadata was removed.
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
names(post_ABU) <- c('alpha', 'f', 'theta')

pred_ABU <-
  lapply(1:7, FUN =
           function(i) {

             indx <- which(i == dat$Site)[1]
             S <- phenology$Site[indx]

             site <-
               lapply(1:12, FUN =
                        function(j) {

                          GP <- paste0('f[', j, ',', i, ']')
                          t <- paste0('theta[', i, ']')

                          cod <- paste0(S, '_', j)
                          indx_SEff <- which(cod == phenology$cod)
                          indx_months <- which(i == phenology$month)

                          est <-
                            with(post_ABU,
                                 {
                                   alpha[, 1, drop = T] +
                                     f[, GP, drop = T] +
                                     theta[, t, drop = T]
                                 })

                          if (length(indx_SEff) > 0) {
                            est <- est + dat$SampEff_std[indx_SEff]
                          } else {
                            est <- est + median(dat$SampEff_std[indx_months])
                          }

                          est <-
                            tibble(Site = i,
                                   date = dmy(paste0('01-', j, '-2025')),
                                   ABU_mu = sample(est, 1e3))
                          est
                        })

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

             site$Site <- S
             site
           })

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

est_ABU <-
  lapply(1:7, FUN =
           function(i) {
             indx <- which(i == dat$Site)[1]
             S <- phenology$Site[indx]

             site <-
               lapply(1:12, FUN =
                        function(j) {

                          GP <- paste0('f[', j, ',', i, ']')
                          t <- paste0('theta[', i, ']')

                          cod <- paste0(S, '_', j)
                          indx_SEff <- which(cod == phenology$cod)
                          indx_months <- which(i == phenology$month)

                          est <-
                            with(post_ABU,
                                 {
                                   alpha[, 1, drop = T] +
                                     f[, GP, drop = T] +
                                     theta[, t, drop = T]
                                 })

                          if (length(indx_SEff) > 0) {
                            est <- est + dat$SampEff_std[indx_SEff]
                          } else {
                            est <- est + median(dat$SampEff_std[indx_months])
                          }

                          est <-
                            tibble(Site = i,
                                   date = dmy(paste0('01-', j, '-2025')),
                                   ABU_mu = mean(est),
                                   ABU_li = quantile(est, 0.025),
                                   ABU_ls = quantile(est, 0.975))
                          est
                        })

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

             site$Site <- S
             site
           })

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

S5.6 Plotting posterior distribution

```
ggplot() +
  geom_line(data = phenology, aes(date, ABU_mu, color = 'No imputation')) +
  geom_point(data = phenology, aes(date, ABU_mu, color = 'No imputation'),
             size = 1.5) +
  geom_ribbon(data = phenology,
              aes(date, ABU_mu, ymin = ABU_li, ymax = ABU_ls,
                  fill = 'No imputation'),
              alpha = 0.3) +
  geom_line(data = est_ABU, aes(date, ABU_mu, color = 'Imputation'), lwd = 0.1) +
  geom_point(data = est_ABU, aes(date, ABU_mu, color = 'Imputation'), size = 0.2) +
  geom_ribbon(data = est_ABU,
              aes(date, ABU_mu, ymin = ABU_li, ymax = ABU_ls, fill = 'Imputation'),
              alpha = 0.3) +
  scale_x_date(date_labels = '%b', date_breaks = '2 month') +
  facet_wrap(~Site) +
  labs(y = 'Fruit abundance') +
  theme_bw()
```

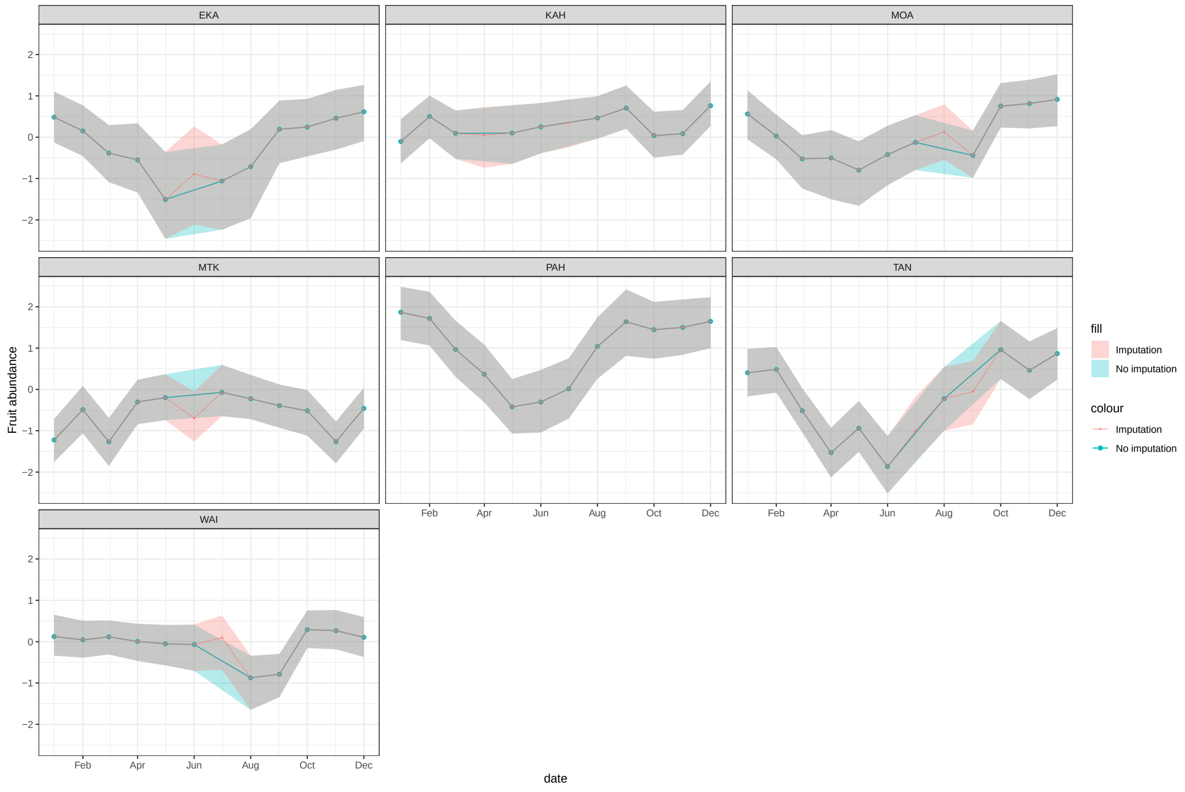

Figure S6: Monthly species richnes of fruiting plants in seven sites at the Oahu island. Non-overlapped red dotes indicates imputed months

```
est_ABU$month <- month(est_ABU$date)
est_S$month <- month(est_S$date)

# data for global models

# saveRDS(list(abun_phenology = est_ABU,
#              S_phenology = est_S), 'phenology_data.rds')
#
#
# saveRDS(list(abun_phenology = pred_ABU,
#              S_phenology = pred_S), 'phenology_data_PREDICTED.rds')
```

S6 Computational environment

```r
sessionInfo()
```

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

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

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

time zone: America/Sao_Paulo
tzcode source: internal

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

other attached packages:
 [1] tidyr_1.3.1         patchwork_1.3.1     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      forcats_1.0.0      
[10] lubridate_1.9.4     ggplot2_3.5.2       dplyr_1.1.4        

loaded via a namespace (and not attached):
 [1] tensorA_0.36.2.1     generics_0.1.4       shape_1.4.6.1       
 [4] lattice_0.22-7       extrafontdb_1.0      digest_0.6.37       
 [7] evaluate_1.0.4       grid_4.5.1           timechange_0.3.0    
[10] RColorBrewer_1.1-3   mvtnorm_1.3-3        fastmap_1.2.0       
[13] cellranger_1.1.0     jsonlite_2.0.0       processx_3.8.6      
[16] backports_1.5.0      ps_1.9.1             purrr_1.0.4         
[19] scales_1.4.0         abind_1.4-8          cli_3.6.5           
[22] rlang_1.1.6          withr_3.0.2          yaml_2.3.10         
[25] tools_4.5.1          checkmate_2.3.2      coda_0.19-4.1       
[28] vctrs_0.6.5          R6_2.6.1             matrixStats_1.5.0   
[31] lifecycle_1.0.4      MASS_7.3-65          pkgconfig_2.0.3     
[34] pillar_1.11.0        gtable_0.3.6         data.table_1.17.8   
```

```
[37] loo_2.8.0            glue_1.8.0           xfun_0.52
[40] tibble_3.3.0         tidyselect_1.2.1     rstudioapi_0.17.1
[43] knitr_1.50           extrafont_0.19       farver_2.1.2
[46] htmltools_0.5.8.1    labeling_0.4.3       rmarkdown_2.29
[49] Rttf2pt1_1.3.12      compiler_4.5.1       distributional_0.5.0
```

S7 Cited literature

- Vizentin-Bugoni, J., Sperry, J. H., Patrick Kelley, J., Gleditsch, J. M., Foster, J. T., Drake, D. R., Hruska, A. M., Wilcox, R. C., Case, S. B., Tarwater, C. E., & designed research, C. (2021). Ecological correlates of species' roles in highly invaded seed dispersal networks. PNAS 118(4). https://doi.org/10.1073/pnas.2009532118