Eco-evolutionary hypothesis testing with Approximate Bayesian Computing (ABC)

Jelena H. Pantel

2022-09-12

```
library(ecoevor)
library(abc)
library(here)
```

1 Overview

The goal of this vignette is to illustrate how to use Approximate Bayesian Computing (ABC, Beaumont 2010; Csilléry et al. 2012) to (i) assign posterior probabilities to an observed dataset for each of a set of alternative hypothesized models and (ii) estimate unmeasured parameter values for the most likely model to have produced the observed data. For an example observed dataset, we simulate population and trait dynamics in a 3-species model of growth and competition (a Leslie-Gower model, which is a discrete-time Lotka-Volterra model). In this model, we also include the possibility that the population growth rate can evolve (using a model of evolutionary rescue introduced by Gomulkiewicz & Holt 1995). The model is as follows:

$$N_{i,t+1} = \frac{\hat{W} e^{\frac{-[(\frac{w+(1-h^2)P}{P+w})(E-x_t)]^2}{2(P+w)}} N_{i,t}}{1 + \alpha_{ii} N_{i,t} + \sum_j \alpha_{ij} N_{j,t}}$$

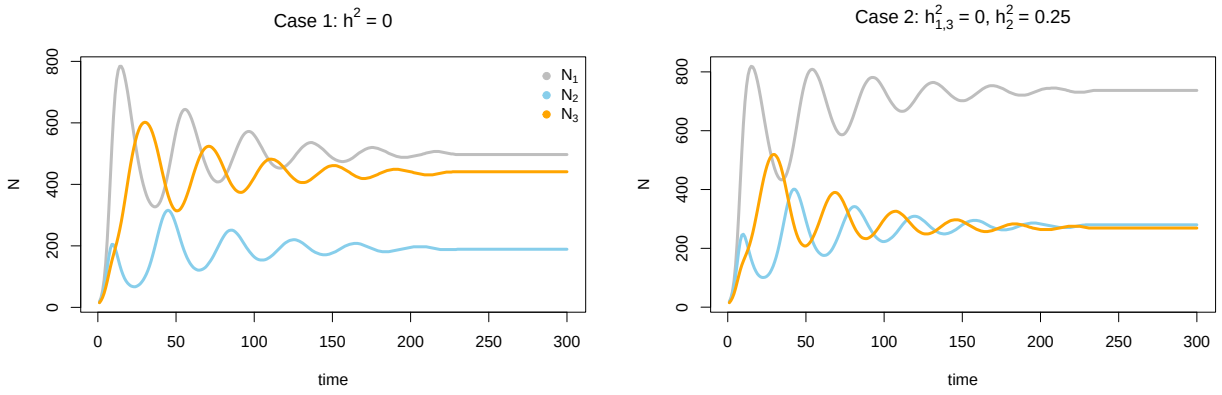
where N_i, t is the population size of a species at time t, $\hat{W}$ is calculated as $\hat{W} = W_{max}\sqrt{(\frac{w}{P+w})}$, W_{max} is the species' maximum per-capita growth rate, w is the width of the Gaussian fitness function (which determines the strength of selection), P is is the width of the distribution of the phenotype x, h^2 is the heritability of the trait x, E is the local environmental optimum trait value, x_t is the trait value of the species at time t, α_{ii} is the intraspecific competition coefficient and α_{ij} is the interspecific competition coefficient.

2 Simulation

The function *lg3_mod* simulates population growth for all 3 species in a single time-step, and the function *LV_evol* simulates population growth for all 3 species across a number of user-specified time-steps. Our goal is to simulate population dynamics for two scenarios, in the absence of trait evolution and with evolution in a trait value for a single species. We will then take the model-generated data from the evolving species model, and use ABC to (i) determine which hypothesized model produced this data and (ii) estimate the model parameters used to produce this 'observed' data. To imitate a more realistic use-case of comparing observed to model-generated data, instead of using the full time series of population size and evolving trait data, we will use population size data collected every 14 days and trait data from the end of the experiment (t_{300}).

To simulate data under the two scenarios (no evolution, $h^2 = 0$ and evolution, $h_2^2 = 0.25$):

```r
# Representing sampling every 14 days
days <- seq(1, 300, by = 14)

## Case 1: 3 species coexist, none evolve Alphas from
## Gallien et al. 2017, The effects of intransitive
## competition on coexistence
fix_parm.1 <- c(a11 = 0.000666, a12 = 0.000111, a13 = 0.001,
    a21 = 0.001, a22 = 0.000666, a23 = 0.000111, a31 = 0.000111,
    a32 = 0.001, a33 = 0.000666, wmax = 2, w = 0.5, P = 0.025,
    h21 = 0, h22 = 0, h23 = 0)
sim1 <- LV_evol(N0 = c(10, 10, 10), x0 = c(0.4, 0.5, 0.6), E = 0.1,
    time_points = 300, fix_parm.1)
abc_obs_summ.1 <- c(as.vector(sim1[days, c(7:9)]), sim1[300,
    10:12])
matplot(1:300, sim1[, 7:9], type = "l", xlab = "time", ylab = "N",
    col = c("grey", "skyblue", "orange"), lty = 1, lwd = 3, main = substitute(paste("Case 1: ",
        "h"^2, " = 0", sep = "")))
legend(x = "topright", legend = c(expression("N"[1]), expression("N"[2]),
    expression("N"[3])), pch = 19, col = c("grey", "skyblue",
    "orange"), bty = "n")

## Case 2: 3 species coexist, species 2 evolves
fix_parm.2 <- fix_parm.1
fix_parm.2[14] <- 0.25
sim2 <- LV_evol(N0 = c(10, 10, 10), x0 = c(0.4, 0.5, 0.6), E = 0.1,
    time_points = 300, fix_parm.2)
abc_obs_summ.2 <- c(as.vector(sim2[days, c(7:9)]), sim2[300,
    10:12])
matplot(1:300, sim2[, 7:9], type = "l", xlab = "time", ylab = "N",
    col = c("grey", "skyblue", "orange"), lty = 1, lwd = 3, main = substitute(paste("Case 2: ",
        "h"["1,3"]^2, " = 0, ", "h"[2]^2, " = 0.25", sep = "")))
```



3 Fitting observed data to model via ABC

The next goal is to use the function *rand_sim* to generate population and trait time series under two alternative model hypotheses (without and with trait evolution), where parameter values are drawn from prior distributions. The ABC algorithm is as follows:

If the model parameters to be estimated are θ and the observed data is x, then the steps for ABC are:

- (i) sample a candidate parameter vector θ^* from a proposed prior distribution $\pi(\theta)$
- (ii) simulate a dataset x^* from the model described by a conditional probability distribution $f(x|\theta^*)$
- (iii) compare the simulated dataset x^* to the observed data x_0, using a distance function d and a tolerance ε: if $d(x_0, x^*) \leq \varepsilon$, accept θ^{**}

Given the computational inefficiency of defining a single distance function $d(x_0, x^*)$ for the entirety of the dataset, it is instead possible to choose a low-dimensional set of summary statistics that capture information about the data. In this instance, we use the population size for all 3 species every 2 weeks and the t_{300} trait values for all 3 species as the summary statistics, that will be compared to model-generated values via ABC.

Step 1. Match alternative hypotheses to simulation models.

We will use Case 2 (where $h^2_{1,3} = 0$ and $h^2_2 = 0.25$) as our "observed" data set. We will use ABC to determine the probability that this observed data was prodcued by either: H1. A model with no trait evolution ($h^2 = 0$) or H2. a model where trait evolution is possible ($h^2 > 0$). These are implemented as model choices in the function *rand_sim*. We will run 100,000 simulations under H1 and H2, which are then used (via ABC) to compare and classify the 'observed' data to one of the two competing hypotheses. Because 100,000 of these simulations can be time-consuming to run, we show the code here for the simulations and for re-arranging the data for the ABC analysis. However, the actual variables themselves are included as an RData object *abc_lv*. If the number of simulations leads to exceeding system memory, reduce *sim_num* below.

```
# To run simulations in their entirety Using beginning and
# end-point trait values
sim_num <- 1e+05
m1 <- rand_sim(n = sim_num, x0 = c(0.4, 0.5, 0.6))
m2 <- rand_sim(n = sim_num, evol = TRUE, x0 = c(0.4, 0.5, 0.6))

### For models with x0 and x300 trait data vector of
### simulation parameter values, dimensions are [mod x n] x
### 15
abc_sim_parm <- rbind(m1$parm_record, m2$parm_record)
## vector of simulation summary statistics, dimensions are
## [mod x n] x 69
dims <- c((dim(m1$pop_record)[1] * 2), 69)
abc_sim_summ <- array(NA, dim = dims, dimnames = list(NULL, c(paste("s1.",
    days, sep = ""), paste("s2.", days, sep = ""), paste("s3.",
    days, sep = ""), "s1.x.300", "s2.x.300", "s3.x.300")))

# Fixed x0
for (i in 1:(dims[1]/2)) {
    abc_sim_summ[i, ] <- c(as.vector(m1$pop_record[i, days, c(7:9)]),
        m1$pop_record[i, 300, 10:12])
}

for (i in ((dims[1]/2) + 1):dims[1]) {
    abc_sim_summ[i, ] <- c(as.vector(m2$pop_record[(i - (dims[1]/2)),
        days, c(7:9)]), m2$pop_record[(i - (dims[1]/2)), 300,
        10:12])
}
rm(list = c("m1", "m2"))

## Using only end-point trait values
```

```
m3 <- rand_sim(n = sim_num)
m4 <- rand_sim(n = sim_num, evol = TRUE)

### For models with only x300 trait data
abc_sim_parm_x0 <- rbind(m3$parm_record, m4$parm_record)
dims <- c((dim(m3$pop_record)[1] * 2), 69)
abc_sim_summ_x0 <- array(NA, dim = dims, dimnames = list(NULL,
    c(paste("s1.", days, sep = ""), paste("s2.", days, sep = ""),
        paste("s3.", days, sep = ""), "s1.x.300", "s2.x.300",
        "s3.x.300")))

# Random x0
for (i in 1:(dims[1]/2)) {
    abc_sim_summ_x0[i, ] <- c(as.vector(m3$pop_record[i, days,
        c(7:9)]), m3$pop_record[i, 300, 10:12])
}

for (i in ((dims[1]/2) + 1):dims[1]) {
    abc_sim_summ_x0[i, ] <- c(as.vector(m4$pop_record[(i - (dims[1]/2)),
        days, c(7:9)]), m4$pop_record[(i - (dims[1]/2)), 300,
        10:12])
}

# Save x0 values for model with evolution (for later
# posterior predictive checks)
x0_rec <- m4$pop_record[, 1, 4:6]

rm(list = c("m3", "m4"))

## vector of simulation models, dimensions are [mod x n] x
## 69
abc_sim_mods <- rep(c("no_evol", "evol"), each = sim_num)

## Save all relevant variables into a list (same as found
## in ecoevoR package data)
abc_lv <- list(abc_sim_parm = abc_sim_parm, abc_sim_parm_x0 = abc_sim_parm_x0,
    abc_sim_summ = abc_sim_summ, abc_sim_summ_x0 = abc_sim_summ_x0,
    abc_sim_mods = abc_sim_mods, x0_rec = x0_rec)
```

Step 2. Prepare ABC

To determine whether the summary statistics are sufficient for identifying the model that generated a dataset, we first use ABC to re-classify the generating model for simulated datasets. We demonstrate this process with (i) a dataset with population size every 14 days and the beginning and end trait values (x_0 and x_{300}) and (ii) a dataset with population size every 14 days and the end trait values (x_{300}). To prepare the ABC, we need a vector of model parameter values (for all simulations across both models) and associated summary statistics. These are generated in the code in Step 1, and those variables were saved in the accompanying dataset *abc_lv.rda*.

```
datapath <- here::here("data")
load(file.path(datapath,"abc_lv.rda"))
list2env(abc_lv,globalenv())
```

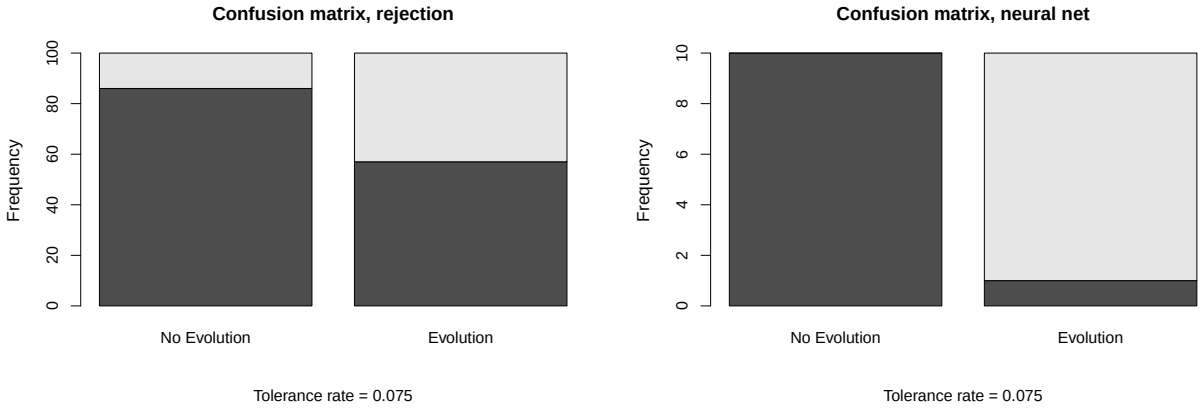
Step 3. Run ABC

To run the ABC, we first use leave-one-out cross-validation (loocv) to determine whether the simulation models and their produced summary statistics are sufficient to accurately classify different simulation-produced summart datasets, using the *abc* package. We show here the results for a rejection method with the x_0 and x_{300} trait values, and then we show the difference between a rejection and neuralnet method for the test with only the t_{300} trait data, using loocv on 100 of the simulated datasets for rejection and 10 for neural net. This also takes a very long time, so we show the code here, but load the results from the *abc_lv* variable as well.

```
cv.modsel <- abc::cv4postpr(abc_sim_mods, abc_sim_summ, nval = 100,
    tol = 0.075, method = "rejection")

cv.modsel.x0 <- abc::cv4postpr(abc_sim_mods, abc_sim_summ_x0,
    nval = 100, tol = 0.075, method = "rejection")

# Neural net
cv.modsel.x0.nn <- abc::cv4postpr(abc_sim_mods, abc_sim_summ_x0,
    nval = 10, tol = 0.075, method = "neuralnet")

abc_lv[[7]] <- cv.modsel
names(abc_lv)[7] <- "cv.modsel"
abc_lv[[8]] <- cv.modsel.x0
names(abc_lv)[8] <- "cv.modsel.x0"
abc_lv[[9]] <- cv.modsel.x0.nn
names(abc_lv)[9] <- "cv.modsel.x0.nn"
```

```
s1 <- summary(cv.modsel)
#> Confusion matrix based on 100 samples for each model.
#>
#> $tol0.075
#>         evol no_evol
#> evol      87      13
#> no_evol   14      86
#>
#>
#> Mean model posterior probabilities (rejection)
#>
#> $tol0.075
#>           evol no_evol
#> evol    0.6649  0.3351
#> no_evol 0.0915  0.9085
s2 <- summary(cv.modsel.x0)
#> Confusion matrix based on 100 samples for each model.
#>
#> $tol0.075
#>         evol no_evol
#> evol      86      14
#> no_evol   57      43
#>
#>
#> Mean model posterior probabilities (rejection)
#>
#> $tol0.075
```

```
#>           evol no_evol
#> evol    0.6376  0.3624
#> no_evol 0.3988  0.6012
s3 <- summary(cv.modsel.x0.nn)
#> Confusion matrix based on 10 samples for each model.
#>
#> $tol0.075
#>         evol no_evol
#> evol      10       0
#> no_evol    1       9
#>
#>
#> Mean model posterior probabilities (neuralnet)
#>
#> $tol0.075
#>           evol no_evol
#> evol    0.9607  0.0393
#> no_evol 0.1048  0.8952
plot(cv.modsel.x0, names.arg=c("No Evolution", "Evolution"), caption="Confusion matrix, rejection")
plot(cv.modsel.x0.nn, names.arg=c("No Evolution", "Evolution"), caption="Confusion matrix, neural net")
```



We can see here that the neural net is better able to classify randomly chosen simulations (n=10, for computational efficiency).

Now we can use these simulations to classify the observed dataset (generated in Step 1 as Case 2 where $h^2_{1,3} = 0$ and $h^2_2 = 0.25$), generating posterior probabilities for each of the two alternative hypotheses (H1: no evolution, H2: evolution). We first compare the rejection and neuralnet methods. We first show results when using a subset of the "observed" data where population size data is collected every 14 days and trait data is from the beginning and end of the experiment (t_0 and t_{300}). (note the neural net model takes several minutes)

```
## i. Calculate the posterior probabilities of each of the
## two models Using known x0 and X300
abc_obs_summ <- rbind(abc_obs_summ.1, abc_obs_summ.2)
colnames(abc_obs_summ) <- c(paste("s1.", days, sep = ""), paste("s2.",
    days, sep = ""), paste("s3.", days, sep = ""), "s1.x.300",
    "s2.x.300", "s3.x.300")

# modsel.1a <-
# abc::postpr(abc_obs_summ['abc_obs_summ.1',],abc_sim_mods,abc_sim_summ,tol=0.05,method='rejection')
```

```
# modsel.1b <-
# abc::postpr(abc_obs_summ['abc_obs_summ.1',],abc_sim_mods,abc_sim_summ,tol=0.1,method='neuralnet')
# # Does not work, as all models are perfectly
# re-classified by rejection method #
modsel.2a <- abc::postpr(abc_obs_summ["abc_obs_summ.2", ], abc_sim_mods,
    abc_sim_summ, tol = 0.05, method = "rejection")
modsel.2b <- abc::postpr(abc_obs_summ["abc_obs_summ.2", ], abc_sim_mods,
    abc_sim_summ, tol = 0.05, method = "neuralnet")
```

When we include the summary statistics for our observed dataset of population size data collected every 14 days and trait data from the beginning and end of the experiment (t_0 and t_{300}), we can see that the neural network model performed well. The posterior probability that the observed summary statistics were produced by a model with no trait evolution (untrue) is: P(H1: $h^2 = 0$) = 0.0309183, and by a model with trait evolution (true) is: P(H2: $h^2 > 0$) = 0.9690817.

However, that was the dataset where the traits are known at both t_0 and t_{300}, which contains the clear information that the traits for species 2 change over time ($x_{0,2} = 0.5$ and $x_{300,2} = 0.11$). To observe how ABC performs in model selection when we have less information, we compare our observed data *abc_obs_summ.2* to the summary statistics produced by two alternative models when the initial trait value is unknown, and thus the simulation was run with randomly chosen x_0 trait values). This is a more difficult task, and even the neural network fails to produce a posterior prediction for which model (no evolution, evolution) produced the observed data.

```
## Using random x0 and known x300
modsel.3a <- abc::postpr(abc_obs_summ["abc_obs_summ.1", ], abc_sim_mods,
    abc_sim_summ_x0, tol = 0.05, method = "rejection")
modsel.3b <- abc::postpr(abc_obs_summ["abc_obs_summ.1", ], abc_sim_mods,
    abc_sim_summ_x0, tol = 0.05, method = "neuralnet")
modsel.4a <- abc::postpr(abc_obs_summ["abc_obs_summ.2", ], abc_sim_mods,
    abc_sim_summ_x0, tol = 0.05, method = "rejection")
modsel.4b <- abc::postpr(abc_obs_summ["abc_obs_summ.2", ], abc_sim_mods,
    abc_sim_summ_x0, tol = 0.05, method = "neuralnet")
```

```
m3a <- summary(modsel.3a)
m3b <- summary(modsel.3b)
m4a <- summary(modsel.4a)
m4b <- summary(modsel.4b)
```

```
m3a$Prob
#>    evol no_evol
#>  0.5329  0.4671
m3b$neuralnet$Prob
#>      evol   no_evol
#> 0.6478203 0.3521797
m4a$Prob
#>    evol no_evol
#>  0.5406  0.4594
m4b$neuralnet$Prob
#>      evol   no_evol
#> 0.8802257 0.1197743
```

In this instance when we only consider the end-point x_{300} trait data, we take one additional processing step - we limit comparison of the observed data to only simulation model results where all 3 species persisted, which is the best match to the observed data.

```
## Using only sims with 3 species persisting
sub3_evol_summ_x0 <- abc_sim_summ_x0[abc_sim_summ_x0[, 22] >
    0 & abc_sim_summ_x0[, 44] > 0 & abc_sim_summ_x0[, 66] > 0,
    ]
idx <- which(abc_sim_summ_x0[, 22] > 0 & abc_sim_summ_x0[, 44] >
    0 & abc_sim_summ_x0[, 66] > 0)
sub_abc_sim_mods <- abc_sim_mods[idx]

modsel.5a <- abc::postpr(abc_obs_summ["abc_obs_summ.2", ], sub_abc_sim_mods,
    sub3_evol_summ_x0, tol = 0.05, method = "rejection")
modsel.5b <- abc::postpr(abc_obs_summ["abc_obs_summ.2", ], sub_abc_sim_mods,
    sub3_evol_summ_x0, tol = 0.05, method = "neuralnet")
```

```
m5a <- summary(modsel.5a)
m5b <- summary(modsel.5b)
```

```
m5a$Prob
#>      evol   no_evol
#> 0.6227785 0.3772215
m5b$neuralnet$Prob
#>      evol   no_evol
#> 0.7518379 0.2481621
```

When we include the summary statistics for our observed dataset of population size data collected every 14 days and trait data from *only* the end of the experiment (t_{300}), we can see that the neural network model performed well. The posterior probability that the observed summary statistics were produced by a model with no trait evolution (untrue) is: P(H1: $h^2 = 0$) = 0.2481621, and by a model with trait evolution (true) is: P(H2: $h^2 > 0$) = 0.7518379.

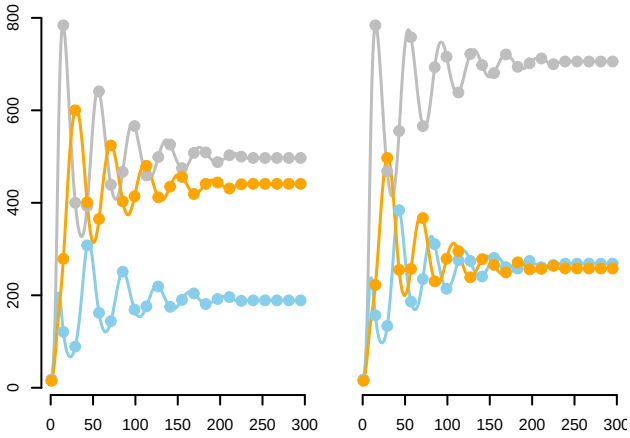
Step 4. Visualizing steps for ABC

Now that we are confident at least one species in the model is evolving (H2 is supported with a high posterior probability), we can estimate posterior distributions for parameter values. To generate realistic posterior distributions, we continue with limiting the simulations to those where all 3 species persisted by the end of the simulation. Here, we walk through the images that appear in Pantel & Becks (in review), Box 2, to explain overall how the ABC process works.

A. Observed data (and observed summary statistics)

```
######## Box2a. Observed summary stats ########
par(mfrow = c(1, 2), mar = c(2, 2, 0.5, 0))
matplot(1:300, sim1[, 7:9], type = "l", xlab = "time", ylab = "N",
    col = c("grey", "skyblue", "orange"), lty = 1, lwd = 3, ann = F,
    axes = F)
axis(1, at = c(0, 50, 100, 150, 200, 250, 300), lwd = 2)
axis(2, at = c(0, 200, 400, 600, 800), lwd = 2)
points(days, sim1[days, 7], pch = 19, col = "grey", cex = 1.5)
points(days, sim1[days, 8], pch = 19, col = "skyblue", cex = 1.5)
points(days, sim1[days, 9], pch = 19, col = "orange", cex = 1.5)

matplot(1:300, sim2[, 7:9], type = "l", xlab = "time", ylab = "N",
    col = c("grey", "skyblue", "orange"), lty = 1, lwd = 3, ann = F,
```

```
    axes = F)
axis(1, at = c(0, 50, 100, 150, 200, 250, 300), lwd = 2)
points(days, sim2[days, 7], pch = 19, col = "grey", cex = 1.5)
points(days, sim2[days, 8], pch = 19, col = "skyblue", cex = 1.5)
points(days, sim2[days, 9], pch = 19, col = "orange", cex = 1.5)
```



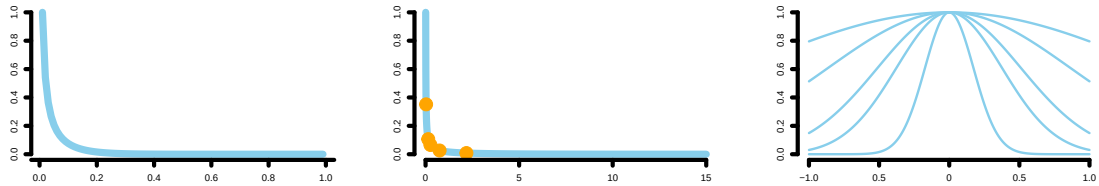
B. Hypothesis testing: generate summary statistics from simulations under alternative hypotheses

(i) competition

(ii) w (strength of selection) and associated fitness functions

```
######## Box2b. Prior parameter distributions ########
######## competition
par(mfrow = c(1, 1))
p = seq(0, 1, length = 100)
db <- dbeta(p, 0.25, 10)
plot(p[2:99], (db[2:99]/max(db[2:99])), ylab = "density", type = "l",
    col = "skyblue", lwd = 12, xlab = " ", ann = F, axes = F)
axis(1, at = c(-0.5, 0, 0.2, 0.4, 0.6, 0.8, 1, 1.5), labels = c("",
    "0.0", "0.2", "0.4", "0.6", "0.8", "1.0", ""), lwd = 7)
axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1), lwd = 7)

## w / strength of selection / width of fitness function
p = seq(0, 15, length = 2000)
dg <- dgamma(p, 0.25, 0.25)
z_range <- seq(-1, 1, 0.001)
plot(p[2:2000], (dg[2:2000]/max(dg[2:2000])), ylab = "density",
    type = "l", col = "skyblue", lwd = 12, xlab = "w", ann = F,
    axes = F)
a <- c(5, 20, 36, 101, 292)
b <- dgamma(p[a], 0.25, 0.25)
points(p[a], (b/max(dg[2:2000])), pch = 19, col = "orange", cex = 3)
axis(1, at = c(0, 5, 10, 15), labels = c("0", "5", "10", "15"),
    lwd = 7)
axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1), lwd = 7)

## Associated fitness functions
fit <- function(z, w2) {
```

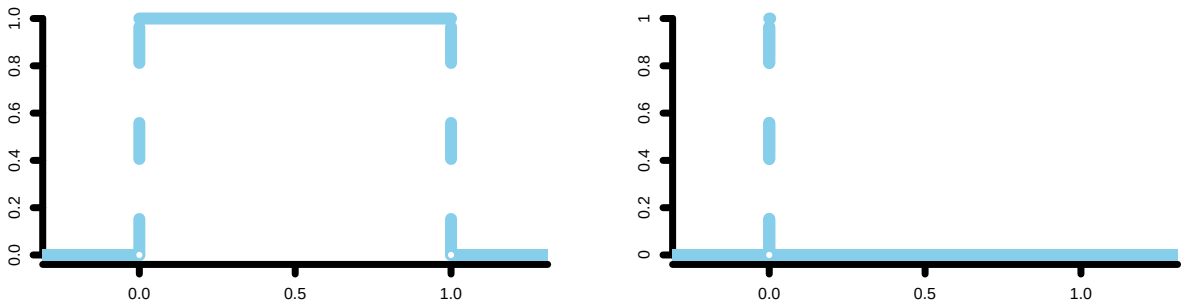
```
    W <- exp((-z^2)/(2 * w2))
}
c <- array(NA, dim = c(length(z_range), length(a)))
for (i in 1:length(a)) {
    for (j in 1:length(z_range)) {
        c[j, i] <- fit(z_range[j], p[a[i]])
    }
}

plot(z_range, c[, 1], type = "l", lwd = 4, col = "skyblue", ylab = "fitness",
    xlab = "trait z", yaxt = "n", ylim = c(0, 1), ann = F, axes = F)
axis(1, at = c(-1, -0.5, 0, 0.5, 1), labels = c("-1.0", "0.5",
    "0", "0.5", "1.0"), lwd = 7)
axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1), lwd = 7)
lines(z_range, c[, 2], type = "l", lwd = 4, col = "skyblue")
lines(z_range, c[, 3], type = "l", lwd = 4, col = "skyblue")
lines(z_range, c[, 4], type = "l", lwd = 4, col = "skyblue")
lines(z_range, c[, 5], type = "l", lwd = 4, col = "skyblue")
```



(iii) heritability

```
### h2
plotunif <- function(x, min = 0, max = 1, lwd = 1, col = 1, ...) {
    # Grid of X-axis values
    if (missing(x)) {
        x <- seq(min - 0.5, max + 0.5, 0.01)
    }

    if (max < min) {
        stop("'min' must be lower than 'max'")
    }

    plot(x, dunif(x, min = min, max = max), xlim = c(min - 0.25,
        max + 0.25), type = "l", lty = 0, ylab = "f(x)", xlab = expression("h"^2),
        ann = F, axes = F)
    axis(1, at = c(-0.5, 0, 0.5, 1, 1.5), labels = c("", "0.0",
        "0.5", "1.0", ""), lwd = 7)
    axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1), lwd = 7)
    segments(min, 1/(max - min), max, 1/(max - min), col = col,
        lwd = lwd)
    segments(min - 2, 0, min, 0, lwd = lwd, col = col)
    segments(max, 0, max + 2, 0, lwd = lwd, col = col)
    points(min, 1/(max - min), pch = 19, col = col)
    points(max, 1/(max - min), pch = 19, col = col)
    segments(min, 0, min, 1/(max - min), lty = 2, col = col,
```

```
        lwd = lwd)
    segments(max, 0, max, 1/(max - min), lty = 2, col = col,
        lwd = lwd)
    points(0, min, pch = 21, col = col, bg = "white")
    points(max, min, pch = 21, col = col, bg = "white")
}
plotunif(min = 0, max = 1, lwd = 12, col = "skyblue", main = "")

# Modified for 0
min <- 0
max <- 1e-14
x <- seq(min - 0.5, max + 0.5, 0.01)

plot(x, dunif(x, min = min, max = max), xlim = c(0 - 0.25, 1 +
    0.25), type = "l", lty = 0, ylab = "f(x)", yaxt = "n", xlab = expression("h"^2),
    ann = F, axes = F)
axis(1, at = c(-0.5, 0, 0.5, 1, 1.5), labels = c("", "0.0", "0.5",
    "1.0", ""), lwd = 7)
axis(2, at = c(0, 2e+13, 4e+13, 6e+13, 8e+13, 1e+14), labels = c(0,
    0.2, 0.4, 0.6, 0.8, 1), lwd = 7)
segments(min, 1/(max - min), max, 1/(max - min), col = "skyblue",
    lwd = 12)
segments(min - 2, 0, min, 0, lwd = 12, col = "skyblue")
segments(max, 0, max + 2, 0, lwd = 12, col = "skyblue")
points(min, 1/(max - min), pch = 19, col = "skyblue")
points(max, 1/(max - min), pch = 19, col = "skyblue")
segments(min, 0, min, 1/(max - min), lty = 2, col = "skyblue",
    lwd = 12)
segments(max, 0, max, 1/(max - min), lty = 2, col = "skyblue",
    lwd = 12)
points(0, min, pch = 21, col = "skyblue", bg = "white")
```

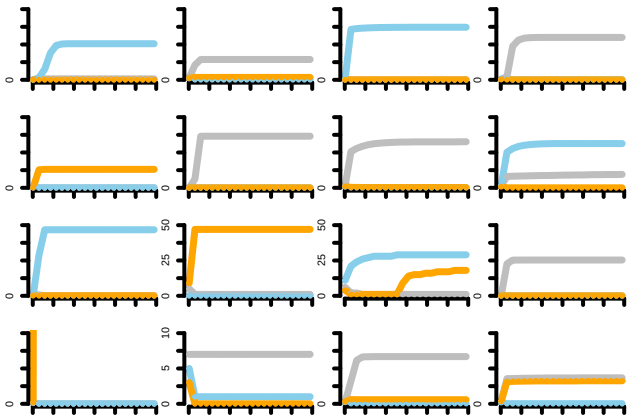


C. Accept or reject summary statistics arising from simulated models, estimate posterior model probability

```
## Euclidean distance between random-parameter value runs
## and observed data
a <- apply(abc_sim_summ_x0, 1, function(x) dist(rbind(abc_obs_summ.2,
    x)))
```

```
# Record of if wach sim was accepted (1) or rejected (0)
# Using a tol=0.075 Return an index for which simulations
# have the 7.5% lowest Euclidean distances from the
# observed summary statistics
accept <- a < quantile(a, probs = 0.075)
index <- which(accept == T)
# In how many did all 3 species persist?
a <- apply(abc_sim_summ_x0[, c(22, 44, 66)], 1, function(x) sum(x >
    0))
# barplot(c(length(a[a ==1]),length(a[a ==2]),length(a[a
# ==3])))
index_3 <- which(a == 3)
## Choose which plots I show for Box1,c choice <-
## c(sort(sample(1:100000,8)),sort(sample(100001:200000,8)))
## choice[3] <-
## sample(setdiff(index_3[index_3<=100000],index[index<=100000]),1)
## choice[5] <-
## sample(intersect(index[index<=100000],index_3[index_3<=100000]),1)
## choice[c(10,16)] <-
## sample(intersect(index[index>100000],index_3[index_3>100000]),2)
## choice[11] <-
## sample(setdiff(index_3[index_3>100000],index[index>100000]),1)
## choice[14] <-
## sample(setdiff(index[index>100000],index_3[index_3>100000]),1)
## choice[c(10,14,16)] <- sample(index[index>100000],1)
choice <- c(7829, 18716, 44946, 57514, 68425, 72639, 85150, 88813,
    105799, 120317, 178858, 147371, 163255, 155317, 171514, 173264)

par(mfcol = c(4, 4), mar = c(2, 2, 0.5, 0))
for (i in 1:16) {
    if (max(abc_sim_summ_x0[choice[i], 1:66]) == 0) {
        lim <- 10
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 10) {
        lim <- 10
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 50) {
        lim <- 50
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 100) {
        lim <- 100
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 500) {
        lim <- 500
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 1000) {
        lim <- 1000
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 5000) {
        lim <- 5000
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 10000) {
        lim <- 10000
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 50000) {
        lim <- 50000
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 1e+05) {
        lim <- 1e+05
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 5e+05) {
        lim <- 5e+05
    } else if (max(abc_sim_summ_x0[choice[i], 1:66]) <= 1e+06) {
```

```
        lim <- 1e+06
    }
    matplot(days, cbind(abc_sim_summ_x0[choice[i], 1:22], abc_sim_summ_x0[choice[i],
        23:44], abc_sim_summ_x0[choice[i], 45:66]), type = "l",
        xlab = "time", ylab = "N", col = c("grey", "skyblue",
            "orange"), lty = 1, ann = F, axes = F, ylim = c(0,
            lim), lwd = 7)
    axis(1, at = c(0, 50, 100, 150, 200, 250, 300), labels = c("",
        "", "", "", "", "", ""), lwd = 4)
    axis(2, at = c(0, quantile(0:lim, probs = c(0.25, 0.5, 0.75)),
        lim), labels = c("0", "", quantile(0:lim, probs = c(0.5)),
        "", lim), lwd = 4)
    points(days, abc_sim_summ_x0[choice[i], 1:22], pch = 19,
        col = "grey")
    points(days, abc_sim_summ_x0[choice[i], 23:44], pch = 19,
        col = "skyblue")
    points(days, abc_sim_summ_x0[choice[i], 45:66], pch = 19,
        col = "orange")
}
```



```
# Note that this value can vary each time the ABC is re-run
m5b$neuralnet$Prob
#>      evol   no_evol
#> 0.7518379 0.2481621
```

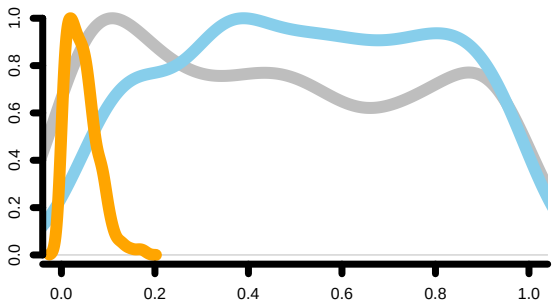
D. Accepted simulation parameters used to estimate posterior distributions of model parameters for observed data

Now that we are confident at least one species in the model is evolving (H2 is supported with a high posterior probability), we can estimate posterior distributions for parameter values. To generate realistic posterior distributions, we limit the simulations to those where all 3 species persisted by the end of the simulation. We consider:

(i) heritability We know the simulation that generated the observed data used $h^2_{1,3} = 0$ and $h^2_2 = 0.25$.

```
######## Box 2d. Model posterior probabilities ########
######## Plots of posterior parameter distributions
######## ######## Using only the 'evol' model, limiting to
######## 3 total species
```

```
sub3_evol_summ_x0 <- abc_sim_summ_x0[abc_sim_mods == "evol" &
    abc_sim_summ_x0[, 22] > 0 & abc_sim_summ_x0[, 44] > 0 & abc_sim_summ_x0[,
    66] > 0, ]
sub3_evol_parm_x0 <- abc_sim_parm_x0[abc_sim_mods == "evol" &
    abc_sim_summ_x0[, 22] > 0 & abc_sim_summ_x0[, 44] > 0 & abc_sim_summ_x0[,
    66] > 0, ]

## heritability
res_h2 <- abc(target = abc_obs_summ["abc_obs_summ.2", ], param = sub3_evol_parm_x0[,
    c("h21", "h22", "h23")], sumstat = sub3_evol_summ_x0, tol = 0.05,
    method = "neuralnet")
# plot(res_h2,param=parm_for_abc[,c('h21','h22','h23')],true=c(0,.25,0))
dscale <- function(den) {
    return(den$y/max(den$y))
}
d1 <- density(res_h2$unadj.values[, 1])
d1$y = dscale(d1)
d2 <- density(res_h2$unadj.values[, 2])
d2$y = dscale(d2)
d3 <- density(res_h2$unadj.values[, 3])
d3$y = dscale(d3)
par(mfrow = c(1, 1))

plot(d1, col = "grey", xlab = expression("h"^2), main = "", xlim = c(0,
    1), lwd = 12, ann = F, axes = F)
lines(d2, col = "skyblue", lwd = 12)
lines(d3, col = "orange", lwd = 12)
axis(1, at = c(-0.5, 0, 0.2, 0.4, 0.6, 0.8, 1, 1.5), labels = c("",
    "0.0", "0.2", "0.4", "0.6", "0.8", "1.0", ""), lwd = 7)
axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1), lwd = 7)
```



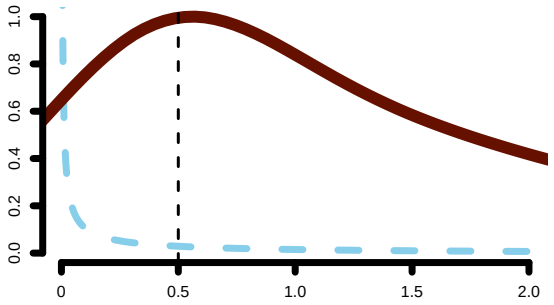
ABC successfully detects that Species 3 (orange) is not evolving, as the posterior probability densoty for h^2 is centered around 0, but struggles to determine the heritability value for Species 1 (grey) and 2 (blue).

(ii) w (strength of selection), and other parameter posterior estimates

Some model parameters might be almost impossible to actually measure in practice, but we can use ABC to estimate them. Here we estimate w, the strength of selection (the width of the Gaussian fitness function, relating trait value x to fitness $\hat{W}$), which was set to 0.25 for the simulation. Here, the prior distribution is shown as the light blue dashed line, the posterior distribution is the dark red line, and the true value is the

black dashed line.

```
## w, strength of selection, and other parameter posterior
## estimates
res_wP <- abc(target = abc_obs_summ["abc_obs_summ.2", ], param = sub3_evol_parm_x0[,
    c("w", "wmax", "P")], sumstat = sub3_evol_summ_x0, tol = 0.05,
    method = "neuralnet")
# plot(res_wP,param=parm_for_abc[,c('w','wmax','P')],true=c(0.5,2,0.025))
# w / strength of selection
p = seq(0, 3, length = 2000)
d <- density(res_wP$unadj.values[, 1])
d$y = dscale(d)

plot(p, (dgamma(p, 0.25, 0.25)/10), ylab = "density", type = "l",
    col = "skyblue", lwd = 7, xlab = "w", xlim = c(0, 2), ylim = c(0,
        1), lty = 2, ann = F, axes = F)
lines(d, col = "#661100", lwd = 12)
abline(v = fix_parm.2["w"], lty = 2, col = "black", lwd = 3)
axis(1, at = c(0, 0.5, 1, 1.5, 2), labels = c("0", "0.5", "1.0",
    "1.5", "2.0"), lwd = 7)
axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1), lwd = 7)
```



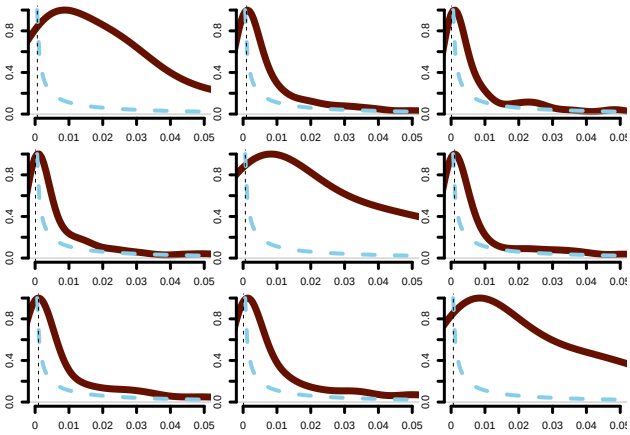
(iii) competition

We know the simulation that generated the observed data had interaction coefficients:

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 0.000666 & 0.001 & 0.000111 \\ 0.000111 & 0.000666 & 0.001 \\ 0.001 & 0.000111 & 0.000666 \end{bmatrix}$$

```
## competition matrix
res_alpha_ii <- abc(target = abc_obs_summ["abc_obs_summ.2", ],
    param = sub3_evol_parm_x0[, c("a11", "a22", "a33")], sumstat = sub3_evol_summ_x0,
    tol = 0.05, method = "neuralnet")
# plot(res_alpha_ii,param=parm_for_abc[,c('a11','a22','a33')],true=c(0.000666,0.000666,0.000666))
res_alpha_ij <- abc(target = abc_obs_summ["abc_obs_summ.2", ],
    param = sub3_evol_parm_x0[, c("a12", "a13", "a21", "a23",
        "a31", "a32")], sumstat = sub3_evol_summ_x0, tol = 0.05,
    method = "neuralnet")
# plot(res_alpha_ij,param=parm_for_abc[,c('a12','a13','a21','a23','a31','a32')],true=c(0.000111,0.00100
```

The posterior distributions for the interaction matrix coefficients $\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} =$



Species interaction matrices are notoriously difficult to estimate. In this instance, intraspecific coefficients were overestimated, but interspecific measures differed from the prior distribution and were quite well estimated.

Recall that we also used unknown (and thus randomly drawn during each simulation) x_0 values. We can estimate posterior distributions for these as well.

```
## initial trait values x0 Subset of all evolution model
## simulations where 3 species persisted, x0_1, x0_2, x0_3
## values
sub3_sim_x0 <- x0_rec[abc_sim_summ_x0[100001:2e+05, 22] > 0 &
    abc_sim_summ_x0[100001:2e+05, 44] > 0 & abc_sim_summ_x0[100001:2e+05,
    66] > 0, ]
# x0 posterior
res_x0 <- abc(target = abc_obs_summ["abc_obs_summ.2", ], param = sub3_sim_x0,
    sumstat = sub3_evol_summ_x0, tol = 0.05, method = "neuralnet")
```

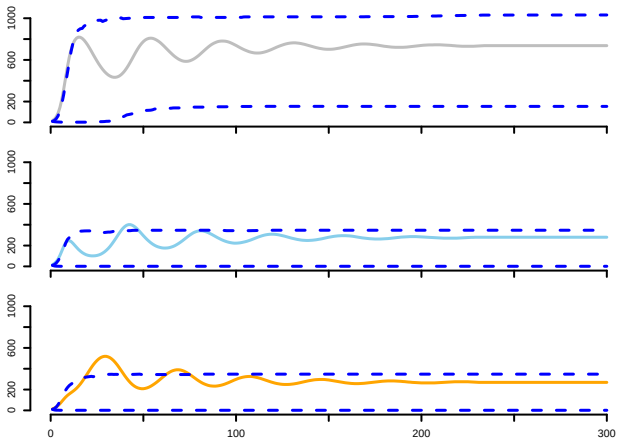
E. Posterior predictive checks

We can confirm the suitability of the model selected for the 'observed' dataset using posterior predictive checks (Rubin, 1984; Gabry et al., 2019). We sample from the ABC marginal posterior distribution, and re-generate simulations. To do this, we identify the simulations that had the lowest Euclidean distance from the observed data.

```
## Restrict simulations to (1) those with all 3 species
## persisting and (2) those least distant from the observed
## values.  Euclidean distance between random-parameter
## value runs and observed data
a <- apply(abc_sim_summ_x0, 1, function(x) dist(rbind(abc_obs_summ.2,
    x)))
## Distance values only for those with 3 species persisting
sub3_a <- a[abc_sim_mods == "evol" & abc_sim_summ_x0[, 22] >
    0 & abc_sim_summ_x0[, 44] > 0 & abc_sim_summ_x0[, 66] > 0]
## Choose the simulations with distance < 3000
idx <- which(sub3_a < 3000)
post_num <- length(idx)
post_parm <- array(NA, dim = c(post_num, 15), dimnames = list(NULL,
```

```
    c("a11", "a12", "a13", "a21", "a22", "a23", "a31", "a32",
        "a33", "wmax", "w", "P", "h21", "h22", "h23")))
post_x0 <- array(NA, dim = c(post_num, 3), dimnames = list(NULL,
    c("x0.1", "x0.2", "x0.3")))
post_parm <- as.data.frame(post_parm)
post_x0 <- as.data.frame(post_x0)
post_sim <- array(NA, dim = c(post_num, 300, 17))

for (i in 1:post_num) {
    post_parm[i, ] <- sub3_evol_parm_x0[idx[i], ]
    post_x0[i, ] <- sub3_sim_x0[idx[i], ]
    post_sim[i, , ] <- LV_evol(N0 = c(10, 10, 10), x0 = as.numeric(post_x0[i,
        ]), E = 0.1, time_points = 300, parms = post_parm[i,
        ])
}
# CI
q1 <- apply(post_sim[, , 1], 2, quantile, probs = c(0.025, 0.957))
q2 <- apply(post_sim[, , 2], 2, quantile, probs = c(0.025, 0.975))
q3 <- apply(post_sim[, , 3], 2, quantile, probs = c(0.025, 0.975))
# Not used: bootstrap CI boot_mean <-
# array(NA,dim=c(1000,300,3)) for(i in 1:300){
# boot_mean[,i,1] <-
# boot::boot(post_sim[,i,1],function(x,i)
# mean(x[i]),R=1000)$t boot_mean[,i,2] <-
# boot::boot(post_sim[,i,2],function(x,i)
# mean(x[i]),R=1000)$t boot_mean[,i,3] <-
# boot::boot(post_sim[,i,3],function(x,i)
# mean(x[i]),R=1000)$t }
par(mfrow = c(3, 1), mar = c(1.9, 2, 0.5, 0))
# plot Species 1 (grey)
matplot(1:300, sim2[, 7], type = "l", xlab = "time", ylab = "N",
    col = c("grey", "skyblue", "orange"), lty = 1, lwd = 3, ann = F,
    axes = F, ylim = c(0, 1010))
axis(1, at = c(0, 50, 100, 150, 200, 250, 300), labels = c("",
    "", "", "", "", "", ""), lwd = 2)
axis(2, at = c(0, 200, 400, 600, 800, 1000), lwd = 2)
lines(1:300, q1[1, ], lty = "dashed", col = "blue", lwd = 3)
lines(1:300, q1[2, ], lty = "dashed", col = "blue", lwd = 3)
# plot Species 2 (blue)
matplot(1:300, sim2[, 8], type = "l", xlab = "time", ylab = "N",
    col = c("skyblue"), lty = 1, lwd = 3, ann = F, axes = F,
    ylim = c(0, 1010))
axis(1, at = c(0, 50, 100, 150, 200, 250, 300), labels = c("",
    "", "", "", "", "", ""), lwd = 2)
axis(2, at = c(0, 200, 400, 600, 800, 1000), lwd = 2)
lines(1:300, q2[1, ], lty = "dashed", col = "blue", lwd = 3)
lines(1:300, q2[2, ], lty = "dashed", col = "blue", lwd = 3)
# plot Species 3 (orange)
matplot(1:300, sim2[, 9], type = "l", xlab = "time", ylab = "N",
    col = c("orange"), lty = 1, lwd = 3, ann = F, axes = F, ylim = c(0,
        1010))
axis(1, at = c(0, 50, 100, 150, 200, 250, 300), labels = c(0,
    "", 100, "", 200, "", 300), lwd = 2)
```

```
axis(2, at = c(0, 200, 400, 600, 800, 1000), lwd = 2)
lines(1:300, q3[1, ], lty = "dashed", col = "blue", lwd = 3)
lines(1:300, q3[2, ], lty = "dashed", col = "blue", lwd = 3)
```



Here, the observed data for each species (Species 1: grey, Species 2: light blue, Species 3: orange) and the 95% confidence interval (dashed blue line) is shown for the subset of accepted simulations that meet our criteria.

References

Beaumont, M. A. Approximate Bayesian Computation in Evolution and Ecology. Annual Review of Ecology, Evolution, and Systematics 41, 379–406 (2010)

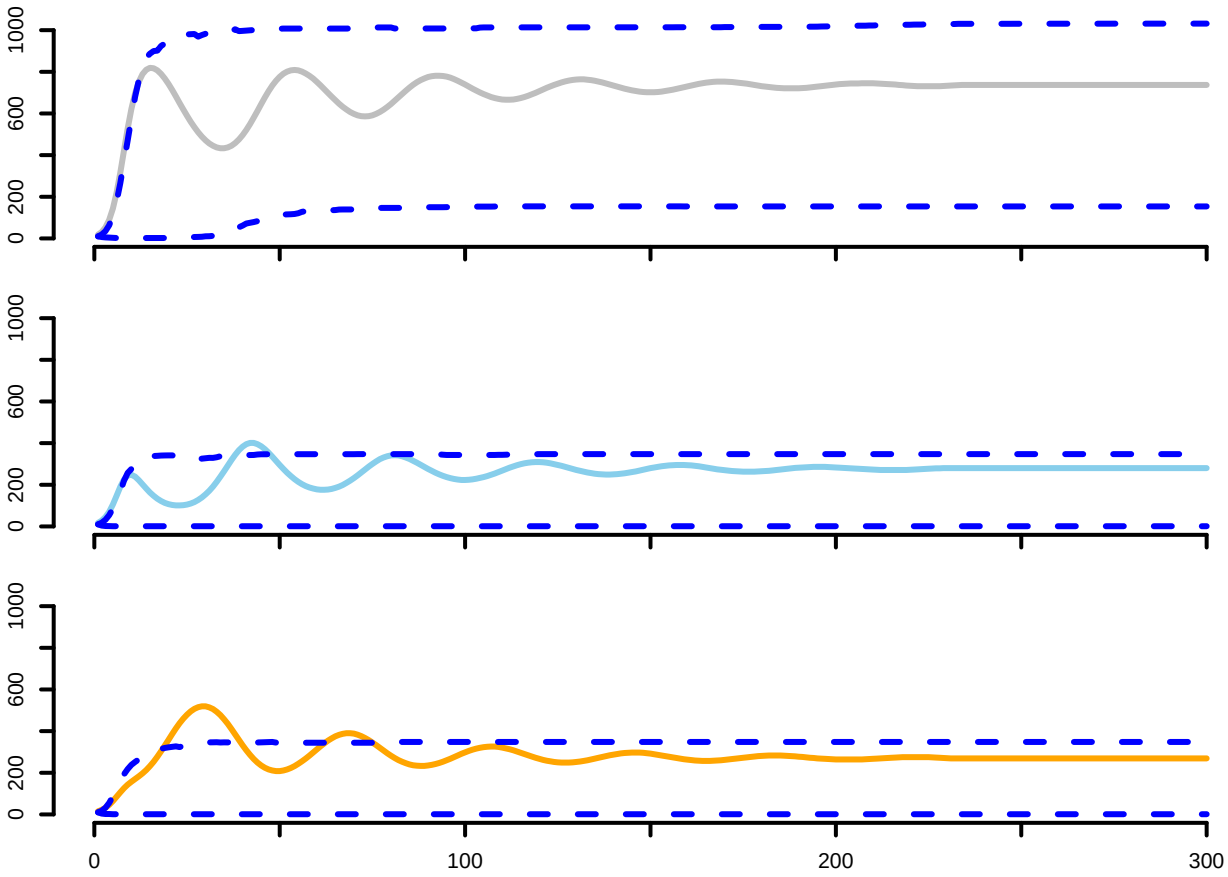
Csilléry, K., François, O. & Blum, M. G. B. abc: an R package for approximate Bayesian computation (ABC). Methods Ecol. Evol. 3, 475–479 (2012)

Gabry, J., D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. 2019. Visualization in Bayesian workflow. Journal of the Royal Statistical Society: Series A (Statistics in Society) 182(2): 389–402.

Gallien, L., Zimmermann, N. E., Levine, J. M., & Adler, P. B. The effects of intransitive competition on coexistence. Ecology Letters, 20(7), 791-800. (2017)

Gomulkiewicz, R. & Holt, R. D. When does evolution by natural selection prevent extinction? Evolution 49, 201–207 (1995)

Rubin, D. B. 1984. Bayesianly justifiable and relevant frequency calculations for the applied statistician. The Annals of Statistics 12(4): 1151–72.


0
200
600
1000
0
200
600
1000
0
200
600
1000
0
100
200
300