ecoevo-ABC

```
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 discrte-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

$$\hat{W} = W_{max} \sqrt(\frac{w}{P+w})$$

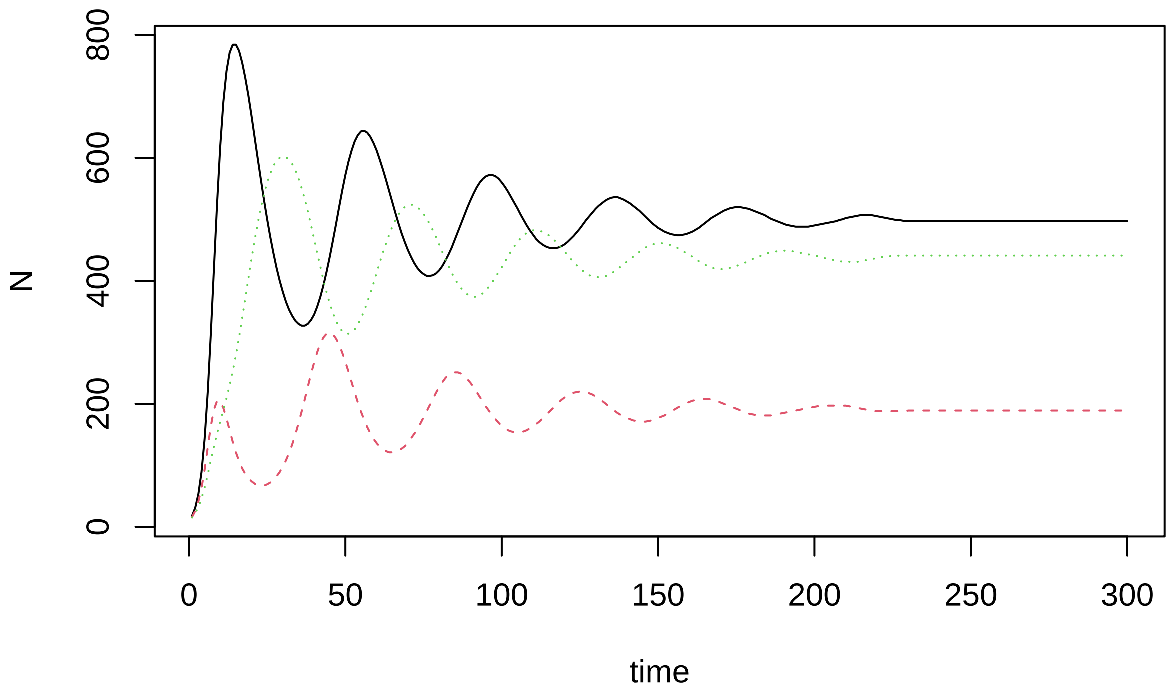
, W_max is the species' maximum per-capita growth rate, w is the width of the Gaussian fitness function (which determines the strength of selection), P is is the width of the distribution of the phenotype x, h^2 is the heritability of the trait x, E is the local environmental optimum trait value, x_t 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

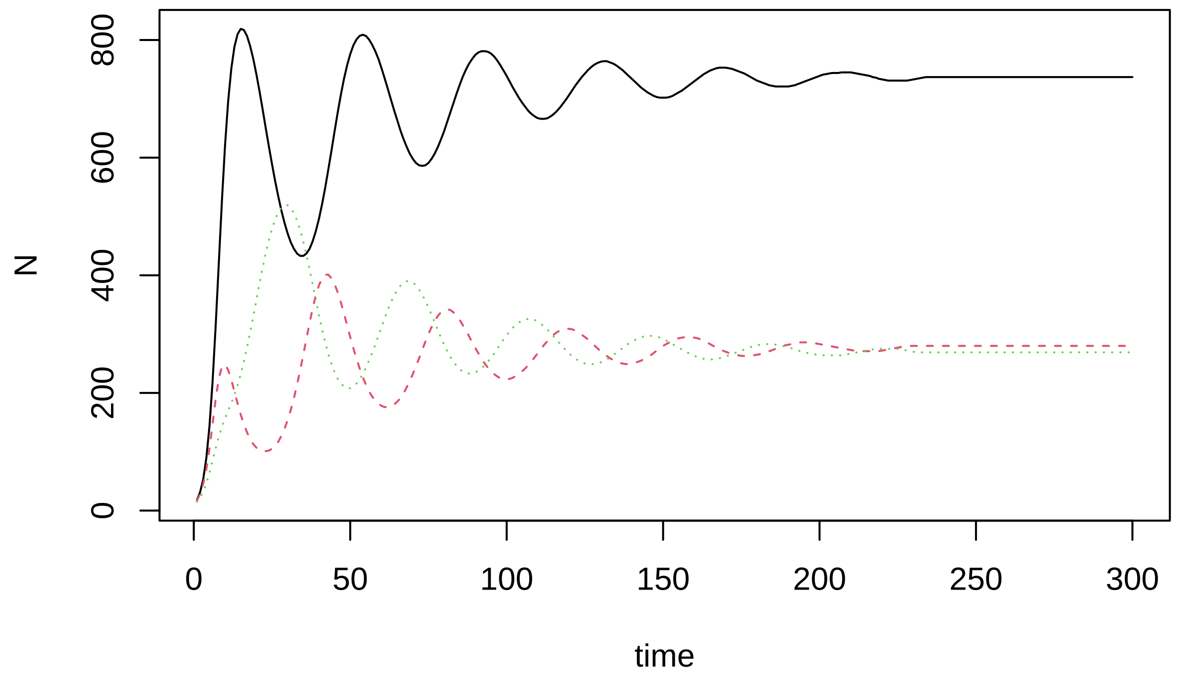
The function *lg3_mod* simulates population growth for all 3 species in a single time-step, and the function *LV_evol* simulates population growth for all 3 species across a number of user-specified time-steps. Our goal is to simulate population dynamics for two scenarios, in the absence of trait evolution 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, $h^2_2 = 0.25$):

```
# 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 = 0.001000,  a21 = 0.001000, a22 = 0.000666, a23 = 
sim1 <- LV_evol(N0=c(10, 10, 10),x0=c(0.4,0.5,0.6),E=.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")
```



```
## 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=.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")
```



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   and the observed data is x, then the steps for ABC are: * (i) sample a candidate parameter vector θ^* from a proposed prior distribution ( ) * (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 : 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.

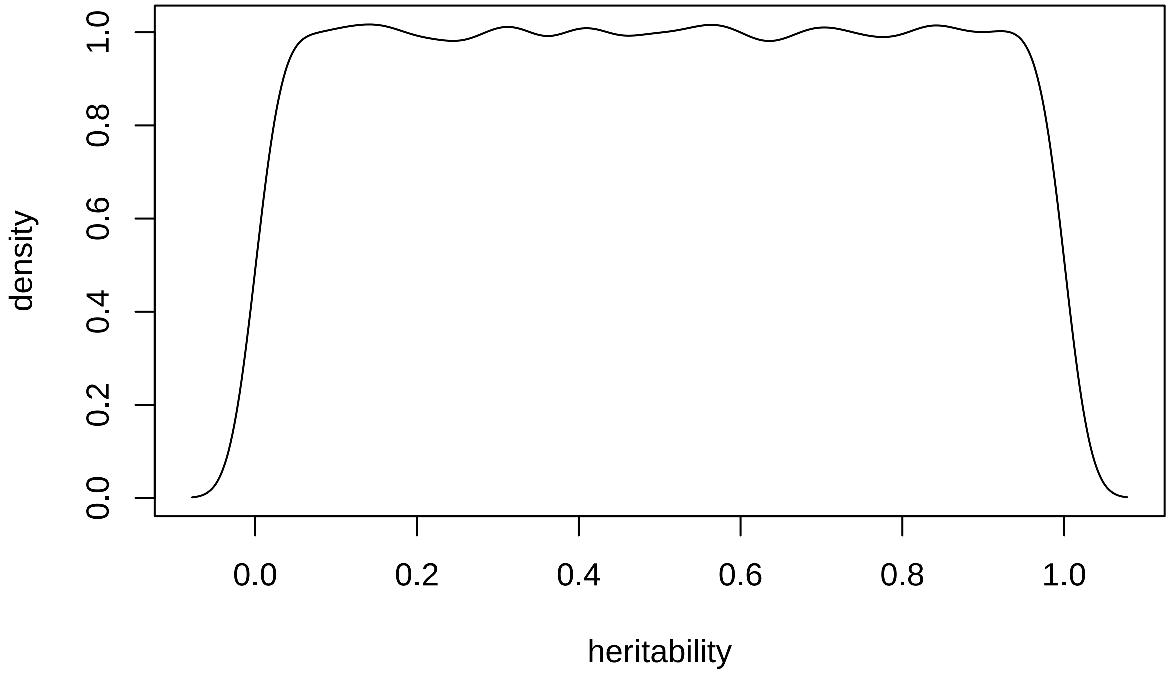
We will use 2 alternative models, where there is no trait evolution ($h^2 = 0$) and where trait evolution is possible ($h^2 > 0$). These are implemented as model choices in the function *rand_sim*. We will use 100,000 simulations. Because these can be time-consuming to run, we show the code here for the simulations and for re-arranging the data for the ABC analysis (Step 2). However, the actual variables themselves are included as an RData object *abc_lv* in the ecoevoR package. If the number of simulations leads to exceeding system memory, reduce *sim_num* below.

```r
# To run simulations in their entirety
## Using beginning and end-point trait values
sim_num <- 100000
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="")

# 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
}
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

# 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 - (
}

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_
rm(list=setdiff(ls(), "abc_lv"))
```

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*.

```
# Plot of distribution of h^2 values for Species 1 across 100,000 simulations with trait evolution
plot(density(abc_lv$abc_sim_parm_x0[100001:200000,13]),main="",xlab="heritability",ylab="density")
```



Step 3. Run ABC

To run the ABC, we first use leave-one-out cross-validation to determine whether the simulation models and their produced summary statistics are sufficient to accurately classify, using the *abc* package. We show here the difference between a rejection and neuralnet method for the test with the end-point only trait data.

```
## Load variables from list into R environment
list2env(abc_lv,globalenv())
## i. Model cross-validation -
cv.modsel <- abc::cv4postpr(abc_sim_mods, abc_sim_summ, nval=5, tol=0.075, method="rejection")
s <- summary(cv.modsel)
plot(cv.modsel, names.arg=c("No Evolution", "Evolution"))
```

```
cv.modsel.x0 <- abc::cv4postpr(abc_sim_mods, abc_sim_summ_x0, nval=5, tol=0.075, method="rejection")
s <- summary(cv.modsel.x0)
plot(cv.modsel.x0, names.arg=c("No Evolution", "Evolution"))

# Neural net
cv.modsel.x0.nn <- abc::cv4postpr(abc_sim_mods, abc_sim_summ_x0, nval=5, tol=0.075, method="neuralnet")
s <- summary(cv.modsel.x0.nn)
plot(cv.modsel.x0.nn, names.arg=c("No Evolution", "Evolution"))

# Add this to list abc_lv that is included with R package data
abc_lv[[6]] <- cv.modsel
names(abc_lv)[7] <- "cv.modsel.x0"
abc_lv[[7]] <- cv.modsel.x0
names(abc_lv)[7] <- "cv.modsel.x0"
abc_lv[[8]] <- cv.modsel.x0.nn
names(abc_lv)[8] <- "cv.modsel.x0.nn"
rm(list=setdiff(ls(), "abc_lv"))
```

Now we can use the simulations to classify the observed datasets (generated in Step 1), generating posterior probabilities for each of the two alternative hypotheses (H1: no evolution, H2: evolution). We first compare the rejection and neuralnet methods, for the data with the known x0 trait value and the unknown x0 trait value.

```
## Load variables from list into R environment
list2env(abc_lv,globalenv())
## 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=""),

modsel.1a <- abc::postpr(abc_obs_summ["abc_obs_summ.1",],abc_sim_mods,abc_sim_summ,tol=0.05,method="rej
modsel.1b <- abc::postpr(abc_obs_summ["abc_obs_summ.1",],abc_sim_mods,abc_sim_summ,tol=0.1,method="neura
modsel.2a <- abc::postpr(abc_obs_summ["abc_obs_summ.2",],abc_sim_mods,abc_sim_summ,tol=0.05,method="rej
modsel.2b <- abc::postpr(abc_obs_summ["abc_obs_summ.2",],abc_sim_mods,abc_sim_summ,tol=0.05,method="neu
```

```
summary(modsel.1a)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.1", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ, tol = 0.05, method = "rejection")
#> Data:
#>  postpr.out$values (10000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>       0       1
#>
#> Bayes factors:
#>         evol no_evol
#> evol               0
#> no_evol  Inf       1
summary(modsel.1b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.1", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ, tol = 0.1, method = "neuralnet")
#> Data:
#>  postpr.out$values (20000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>       0       1
#>
#> Bayes factors:
#>         evol no_evol
#> evol               0
#> no_evol  Inf       1
summary(modsel.2a)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ, tol = 0.05, method = "rejection")
#> Data:
#>  postpr.out$values (10000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.1393  0.8607
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  0.1618
#> no_evol 6.1788  1.0000
summary(modsel.2b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ, tol = 0.05, method = "neuralnet")
#> Data:
#>  postpr.out$values (10000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.1393  0.8607
```

```
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  0.1618
#> no_evol 6.1788  1.0000
#>
#>
#> Posterior model probabilities (neuralnet):
#>    evol no_evol
#>  0.9073  0.0927
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  9.7844
#> no_evol 0.1022  1.0000
## Needed the neuralnet to predict correct model for evolution, but it worked.
```

We can see that the neural network model performed well in estimating the posterior probability that the summary statistics for our observed dataset (abc_obs_summ.2), produced by a model with trait evolution for species 2, was in fact produced by the model with trait evolution ($h^2 > 0$). However, that was the dataset where the known values of the trait at the beginning of the observed period was known (above, `m2 <- rand_sim(n=sim_num,evol=TRUE,x0=c(0.4,0.5,0.6))`, indicating the simulation was run with the known x_0 trait values). To simulate how ABC performs in model selection when we have less information, we compare our observed data *abc_obs_summ.1* to the summary statistics produced by two alternative models when the initial trait value is unknown (above, `m4 <- rand_sim(n=sim_num,evol=TRUE)`, indicating that 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="
modsel.3b <- abc::postpr(abc_obs_summ["abc_obs_summ.1",],abc_sim_mods,abc_sim_summ_x0,tol=0.1,method="n
modsel.4a <- abc::postpr(abc_obs_summ["abc_obs_summ.2",],abc_sim_mods,abc_sim_summ_x0,tol=0.05,method="
modsel.4b <- abc::postpr(abc_obs_summ["abc_obs_summ.2",],abc_sim_mods,abc_sim_summ_x0,tol=0.05,method="
```

```
summary(modsel.3a)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.1", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ_x0, tol = 0.05, method = "rejection")
#> Data:
#>  postpr.out$values (10000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.5329  0.4671
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.1409
#> no_evol 0.8765  1.0000
summary(modsel.3b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.1", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ_x0, tol = 0.1, method = "neuralnet")
#> Data:
#>  postpr.out$values (20000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>   0.534   0.466
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.1459
#> no_evol 0.8727  1.0000
#>
#>
#> Posterior model probabilities (neuralnet):
#>    evol no_evol
#>  0.3576  0.6424
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  0.5566
#> no_evol 1.7967  1.0000
summary(modsel.4a)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ_x0, tol = 0.05, method = "rejection")
#> Data:
#>  postpr.out$values (10000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.5406  0.4594
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.1768
#> no_evol 0.8498  1.0000
summary(modsel.4b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = abc_sim_mods,
#>     sumstat = abc_sim_summ_x0, tol = 0.05, method = "neuralnet")
#> Data:
#>  postpr.out$values (10000 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.5406  0.4594
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.1768
#> no_evol 0.8498  1.0000
#>
#>
#> Posterior model probabilities (neuralnet):
#>    evol no_evol
#>  0.6279  0.3721
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.6874
#> no_evol 0.5926  1.0000
```

In this instance, ABC does well to accurately classify the observed data was produced by the model with evolution when we provide x_0 and x_{300}. When we only consider the end-point x_{300} trait data, ABC cannot clearly distinguish the most likely model.

We take an additional step, where we limit comparison of the observed data to only simulation model results where all 3 species persisted (as is seen in 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
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.1",],sub_abc_sim_mods,sub3_evol_summ_x0,tol=0.05,me
modsel.5b <- abc::postpr(abc_obs_summ["abc_obs_summ.1",],sub_abc_sim_mods,sub3_evol_summ_x0,tol=0.1,met
modsel.6a <- abc::postpr(abc_obs_summ["abc_obs_summ.2",],sub_abc_sim_mods,sub3_evol_summ_x0,tol=0.05,me
modsel.6b <- abc::postpr(abc_obs_summ["abc_obs_summ.2",],sub_abc_sim_mods,sub3_evol_summ_x0,tol=0.05,me
```

```
summary(modsel.5a)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.1", ], index = sub_abc_sim_mods,
#>     sumstat = sub3_evol_summ_x0, tol = 0.05, method = "rejection")
#> Data:
#>  postpr.out$values (889 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#> Warning: Posterior model probabilities are corrected for unequal number
#> of simulations per models.
#>
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>   0.626   0.374
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.3091
#> no_evol 0.7639  1.0000
summary(modsel.5b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.1", ], index = sub_abc_sim_mods,
#>     sumstat = sub3_evol_summ_x0, tol = 0.1, method = "neuralnet")
#> Data:
#>  postpr.out$values (1778 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#> Warning: Posterior model probabilities are corrected for unequal number
#> of simulations per models.
#>
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.6082  0.3918
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.2142
#> no_evol 0.8236  1.0000
#>
#>
#> Posterior model probabilities (neuralnet):
#>    evol no_evol
#>  0.3663  0.6337
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  0.5780
#> no_evol 1.7301  1.0000
summary(modsel.6a)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = sub_abc_sim_mods,
#>     sumstat = sub3_evol_summ_x0, tol = 0.05, method = "rejection")
#> Data:
#>  postpr.out$values (889 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
```

```
#>  evol, no_evol
#> Warning: Posterior model probabilities are corrected for unequal number
#> of simulations per models.
#>
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.6228  0.3772
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.2912
#> no_evol 0.7745  1.0000
summary(modsel.6b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = sub_abc_sim_mods,
#>     sumstat = sub3_evol_summ_x0, tol = 0.05, method = "neuralnet")
#> Data:
#>  postpr.out$values (889 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#> Warning: Posterior model probabilities are corrected for unequal number
#> of simulations per models.
#>
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.6228  0.3772
#>
#> Bayes factors:
#>           evol no_evol
#> evol    1.0000  1.2912
#> no_evol 0.7745  1.0000
#>
#>
#> Posterior model probabilities (neuralnet):
#>    evol no_evol
#>  0.9269  0.0731
#>
#> Bayes factors:
#>            evol no_evol
#> evol     1.0000 12.6784
#> no_evol  0.0789  1.0000
```

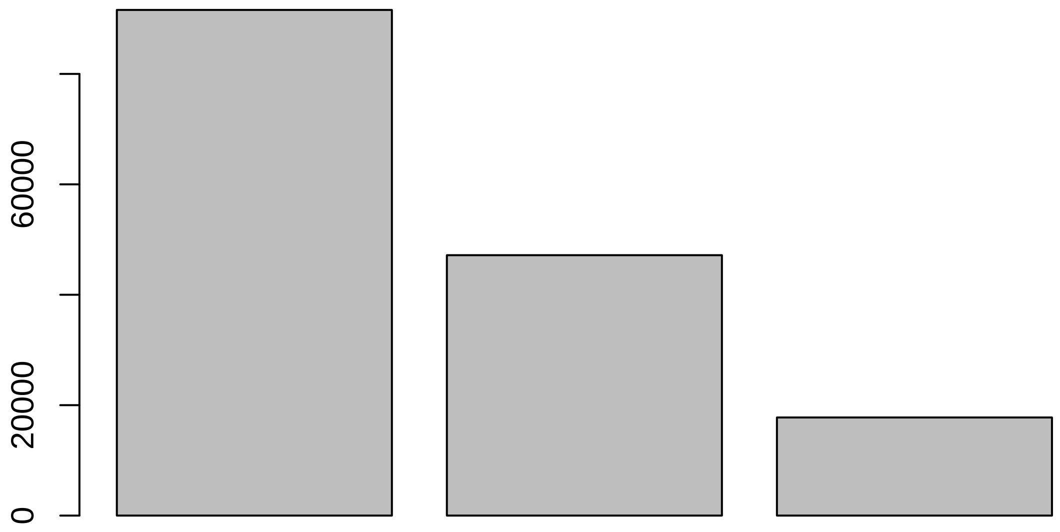
In this instance the neural network performed much better, with a posterior probability of 0.89 that the summary statistics (with only the x_{300} trait values) matched to the model with trait evolution possible ($h^2 > 0$).

4 Visualizing steps for ABC

```r
datapath <- here::here("inst","extdata", "Box1")

######## Box1a. Observed summary stats ########
pdf(file=file.path(datapath,"Box_1a.pdf"),width=12,height=7)
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
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
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)
dev.off()
#> pdf
#>   2

######## Box1b. Prior parameter distributions ########
### 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(-0.5,0.0,0.5,1.0,1.5),labels=c("","0.0","0.5","1.0",""),lwd=7)
  axis(2,at=c(0,0.2,0.4,0.6,0.8,1.0),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")
}
pdf(file=file.path(datapath,"Box_1b.1.pdf"),width=7,height=7)
plotunif(min = 0, max = 1, lwd = 12, col = "skyblue", main = "")
dev.off()
```

```
#> pdf
#>   2

# Modified for 0
min <- 0
max <- 0.00000000000001
x <- seq(min - 0.5, max + 0.5, 0.01)
pdf(file=file.path(datapath,"Box_1b.2.pdf"),width=7,height=7)
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(-0.5,0.0,0.5,1.0,1.5),labels=c("","0.0","0.5","1.0",""),lwd=7)
axis(2,at=c(0e+00,2e+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")
dev.off()
#> pdf
#>   2

## alpha / interaction strength terms
p = seq(0,1, length=100)
db <- dbeta(p, 0.25, 10)
pdf(file=file.path(datapath,"Box_1b.3.pdf"),width=7,height=7)
plot(p[2:99],(db[2:99] / max(db[2:99])), ylab="density", type ="l", col="skyblue", lwd=12, xlab =" ", an
axis(1,at=c(-0.5,0.0,0.2,0.4,0.6,0.8,1.0,1.5),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)
dev.off()
#> pdf
#>   2

## 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,.001)
pdf(file=file.path(datapath,"Box_1b.4.pdf"),width=7,height=7)
plot(p[2:2000], (dg[2:2000]/max(dg[2:2000])), ylab="density", type ="l", col="skyblue", lwd=12, xlab ="
a <- c(5, 20, 36, 101, 292)
b <- dgamma(p[a],.25,.25)
#292, 445)
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)
dev.off()
#> pdf
#>   2

# Associated ftness 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]])
  }
}
pdf(file=file.path(datapath,"Box_1b.5.pdf"),width=7,height=7)
plot(z_range,c[,1],type="l",lwd=4,col="skyblue",ylab="fitness",xlab="trait z",yaxt="n",ylim=c(0,1), ann
axis(1,at=c(-1.0,-0.5,0,0.5,1.0),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")
dev.off()
#> pdf
#>   2

######## Figure 2c. Simulated summary stats with euclidean distances and acceptance / rejection ABC ###
## 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 summ
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 2c
# 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

pdf(file=file.path(datapath,"Box_1c.pdf"),width=14,height=14)
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]) <= 100000){
    lim <- 100000
  } else if (max(abc_sim_summ_x0[choice[i],1:66]) <= 500000){
    lim <- 500000
  } else if (max(abc_sim_summ_x0[choice[i],1:66]) <= 1000000){
    lim <- 1000000
  }
  matplot(days,cbind(abc_sim_summ_x0[choice[i],1:22],abc_sim_summ_x0[choice[i],23:44],abc_sim_summ_x0[c
  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(.25,.5,.75)),lim),labels=c("0","",quantile(0:lim,probs=c(.5)),""
  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")
}
dev.off()
#> pdf
#>   2

######## Figure 2d. Model posterior probabilities ########
# Note that this value can vary each time the ABC is re-run
summary(modsel.6b)
#> Call:
#> abc::postpr(target = abc_obs_summ["abc_obs_summ.2", ], index = sub_abc_sim_mods,
#>     sumstat = sub3_evol_summ_x0, tol = 0.05, method = "neuralnet")
#> Data:
#>  postpr.out$values (889 posterior samples)
#> Models a priori:
#>  evol, no_evol
#> Models a posteriori:
#>  evol, no_evol
#> Warning: Posterior model probabilities are corrected for unequal number
#> of simulations per models.
#>
#>
#> Proportion of accepted simulations (rejection):
#>    evol no_evol
#>  0.6228  0.3772
#>
#> Bayes factors:
#>            evol no_evol
#> evol    1.0000  1.2912
#> no_evol 0.7745  1.0000
#>
#>
#> Posterior model probabilities (neuralnet):
#>    evol no_evol
#>  0.9269  0.0731
```

```
#>
#> Bayes factors:
#>             evol no_evol
#> evol      1.0000 12.6784
#> no_evol   0.0789  1.0000

######## Figure 2e. 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_summ_x0[,22] > 0 & abc_sim_summ_x0[,44] > 0 & abc_sim_summ

sub3_evol_parm_x0 <- abc_sim_parm_x0[abc_sim_summ_x0[,22] > 0 & abc_sim_summ_x0[,44] > 0 & abc_sim_summ

summ_for_abc <- sub3_evol_summ_x0[sub_abc_sim_mods=="evol",]
parm_for_abc <- sub3_evol_parm_x0[sub_abc_sim_mods=="evol",]

## heritability
res_h2 <- abc(target=abc_obs_summ["abc_obs_summ.2",],param=parm_for_abc[,c("h21","h22","h23")],sumstat=
#> Warning: All parameters are "none" transformed.
#> 12345678910
#> 12345678910
#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)

pdf(file=file.path(datapath,"Box_1d.1.pdf"),width=7,height=7)
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(-0.5,0.0,0.2,0.4,0.6,0.8,1.0,1.5),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)
dev.off()
#> pdf
#>   2

## competition matrix
res_alpha_ii <- abc(target=abc_obs_summ["abc_obs_summ.2",], param=parm_for_abc[,c("a11","a22","a33")],su
#> Warning: All parameters are "none" transformed.
#> 12345678910
#> 12345678910
# 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=parm_for_abc[,c("a12","a13","a21","a2
#> Warning: All parameters are "none" transformed.
#> 12345678910
#> 12345678910
# plot(res_alpha_ij,param=parm_for_abc[,c("a12","a13","a21","a23","a31","a32")],true=c(0.000111,0.00100
#Sp 1
```

```
d11 <- density(res_alpha_ii$unadj.values[,"a11"])
d11$y = dscale(d11)
d12 <- density(res_alpha_ij$unadj.values[,"a12"])
d12$y = dscale(d12)
d13 <- density(res_alpha_ij$unadj.values[,"a13"])
d13$y = dscale(d13)
#Sp 2
d22 <- density(res_alpha_ii$unadj.values[,"a22"])
d22$y = dscale(d22)
d21 <- density(res_alpha_ij$unadj.values[,"a21"])
d21$y = dscale(d21)
d23 <- density(res_alpha_ij$unadj.values[,"a23"])
d23$y = dscale(d23)
#Sp 3
d33 <- density(res_alpha_ii$unadj.values[,"a33"])
d33$y = dscale(d33)
d31 <- density(res_alpha_ij$unadj.values[,"a31"])
d31$y = dscale(d31)
d32 <- density(res_alpha_ij$unadj.values[,"a32"])
d32$y = dscale(d32)

ord1 <- c("11","12","13","21","22","23","31","32","33")
ord2 <- c(expression(" "[11]),expression(" "[12]),expression(" "[13]),expression(" "[21]),expression(" "[2
p = seq(0,0.06, length=100)
db <- dbeta(p, 0.25, 10)

pdf(file=file.path(datapath,"Box_1d.2.pdf"),width=12,height=12)
par(mfcol=c(3,3),mar = c(2, 2, 0.5, 0))
for(i in 1:9){
  plot(get(paste("d",ord1[i],sep="")),,col="#661100",main="",xlim=c(0,0.05),lwd=7,ann=F, axes=F)
  axis(1,at=c(0.0,0.01,0.02,0.03,0.04,0.05),labels=c("0","0.01","0.02","0.03","0.04","0.05"),lwd=3)
  axis(2,at=c(0,0.2,0.4,0.6,0.8,1),lwd=3)

  lines(p[2:99],(db[2:99] / max(db[2:99])),lwd=4,lty=2,col="skyblue")
  abline(v=fix_parm.2[paste("a",ord1[i],sep="")],lty=2,col="black")
}
dev.off()
#> pdf
#>   2

## w, strength of selection, and other parameter posterior estimates
res_wP <- abc(target=abc_obs_summ["abc_obs_summ.2",],param=parm_for_abc[,c("w","wmax","P")],sumstat=summ
#> Warning: All parameters are "none" transformed.
#> 12345678910
#> 12345678910
# 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)
pdf(file=file.path(datapath,"Box_1d.3.pdf"),width=7,height=7)
plot(p, (dgamma(p, 0.25, 0.25)/10), ylab="density", type ="l", col="skyblue", lwd=7, xlab ="w",xlim=c(0
lines(d,col="#661100",lwd=12)
```

```
abline(v=fix_parm.2["w"],lty=2,col="black",lwd=3)
axis(1,at=c(0.0,0.5,1.0,1.5,2.0),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)
dev.off()
#> pdf
#>   2
```

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)

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)