Eco-evolutionary hypothesis testing with Machine Learning (ML)

Jelena H. Pantel

2022-09-12

```
library(ecoevor)
library(here)
library(Boruta)
library(randomForest)
library(caTools)
```

1 Overview

The goal of this vignette is to illustrate how to use machine learning (more specifically a random forest algorithm; Breiman 2001; Liaw & Wiener 2002) to classify an observed feature set (in this instance summary statistics from a simulated 'observed data set'), i.e. assign posterior probabilities to an observed dataset for each of a set of alternative hypothesized models.

For an example observed dataset, use the same simulation of 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, with potentially evolving population growth rate) that is the focus of the ecoevo-ABC vignette (Pantel & Becks 202x, Box 2).

There are 2 key differences from the ABC version presented in Pantel & Becks (2022x) Box 2 / the vignette ecoevo-ABC - first, we use as an observed dataset a coarse time series population size every 14 days and the beginning and end trait values (x_1,x_{300}) (note that the x_1 values aren't included as features, but they are nevertheless known because they were supplied to the simulatin command *LV_evol* - see full ecoevo-ML.rmd code to confirm this). Second, the random forest does not perform as well as ABC for this model + dataset. We leave it in as an example of how ML can be used for eco-evolutionary hypothesis testing in R.

2 Prepare ML

To determine how the summary statistics contribute to identifying the model that generated a dataset, we first use feature selection via a *boruta* algorithm (Kursa & Rudnicki 2010). We focus on the simulated dataset produced in the vignette ecoevo-ABC, a dataset with population size every 14 days and the end trait values (x_{300}). We use the same vector of model parameter values and associated summary statistics saved in the accompanying dataset *abc_lv.rda*.

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

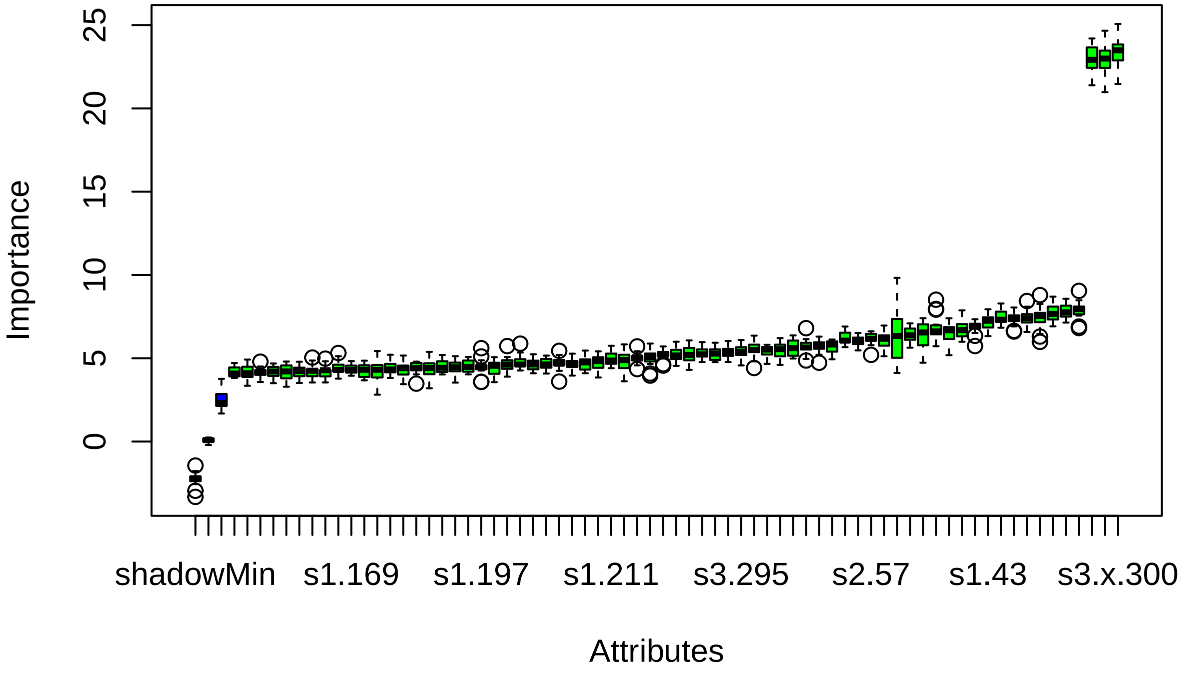
3 Fitting observed data to model via ABC

Step 1. Perform initial feature selection procedure

This step is quite lengthy, so we show the code here, but call the saved object *rf_lv.rda* for subsequent use.

```
set.seed(777)
bor <- Boruta::Boruta(abc_sim_summ, as.factor(abc_sim_mods),
    doTrace = 2, ntree = 500)
```

```
load(file.path(datapath, "rf_lv.rda"))
list2env(rf_lv, globalenv())
plot(bor)
```



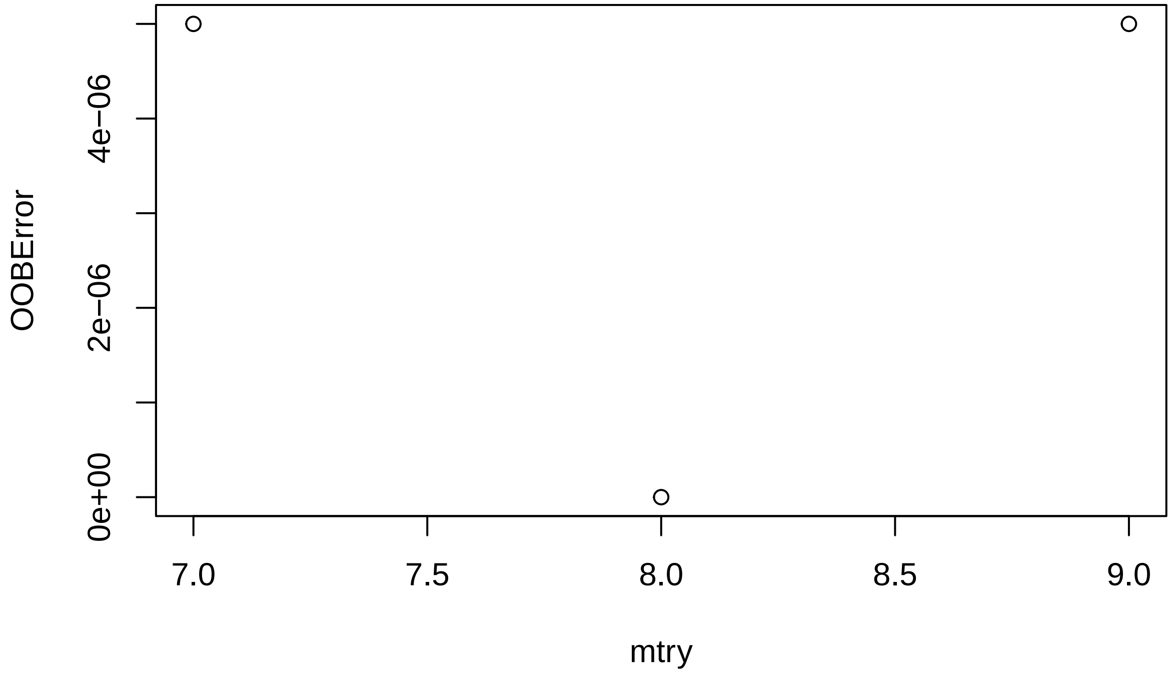
Interestingly, we see that the trait features are most important.

Step 2. Tune hyperparameters for model

Use *tuneRF* to search for an optimal value of *mtry* given the data. *mtry* is a parameter in the random forest algorithm that determines the number of variables randomly sampled as candidates at each split, and the optimal values reduces the 'out-of-bag' score (where random sub-samples of the original training data are taken, random forest models are generated using these subsets, and the prediction error for the remaining "out-of-bag" samples are calculated; Pichler & Hartig 2022). Again, this step can be lengthy, so we load this from the *rf_lv.rda* environment.

```
# improve value can be raised, in our case there isn't much
# difference across mtry values
tune <- randomForest::tuneRF(abc_sim_summ, as.factor(abc_sim_mods),
    stepFactor = 1.2, improve = 0.001, trace = T, plot = T, mtryStart = 8,
    nodesize = 5)

# We also show the code for an alternative attempt to tune
# mtry, which did not change our results tune <-
# tuneRF(abc_sim_summ,as.factor(abc_sim_mods),ntreeTry=500,mtryStart=4,stepFactor=1.5,improve=0.01)
# #
```



In this instance changing the mtry (Number of variables randomly sampled as candidates at each split) doesn't impact the "Out-of-Bag" error estimate. Bagging uses subsampling with replacement to create training samples for the model to learn from. OOB error is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

Step 3. Train ML classifier on the simulations using the summary statistics as features and the model (No-Evol, Evol) as the target variable

```
fac_sim_mod <- factor(abc_sim_mods)

sample <- caTools::sample.split(fac_sim_mod, SplitRatio = 0.75)
train_y <- subset(fac_sim_mod, sample == TRUE)
train_x <- subset(abc_sim_summ, sample == TRUE)
```

```
test_y <- subset(fac_sim_mod, sample == FALSE)
test_x <- subset(abc_sim_summ, sample == FALSE)

tune <- as.data.frame(tune)
m <- tune$mtry[tune$OOBError == min(tune$OOBError, na.rm = TRUE)]
```

```
rf <- randomForest(train_x, train_y, mtry = m)
rf_lv <- list(bor = bor, tune = tune, rf = rf)
save(rf_lv, file = "data/rf_lv.rda")
```

```
rf
#> 
#> Call:
#>  randomForest(x = train_x, y = train_y, mtry = m) 
#>                Type of random forest: classification
#>                      Number of trees: 500
#> No. of variables tried at each split: 8
#> 
#>         OOB estimate of  error rate: 0%
#> Confusion matrix:
#>          evol no_evol class.error
#> evol    75000       0           0
#> no_evol     0   75000           0
```

The confusion matrix indicates a 100% classification accuracy. We now try a test dataset, and again it accurately classifies 100%:

```
pred <- predict(rf, newdata = test_x)
cm <- table(test_y, pred)
cm
#>          pred
#> test_y      evol no_evol
#>   evol     25000       0
#>   no_evol      0   25000
```

Step 4. Posterior model probabilities for 2 'observed' datasets

```
pred_obs <- predict(rf, newdata = abc_obs_summ, type = "prob")
pred_obs
#>                 evol no_evol
#> abc_obs_summ.1 0.006   0.994
#> abc_obs_summ.2 0.354   0.646
#> attr(,"class")
#> [1] "matrix" "array"  "votes"
```

The model prediction is incorrect for the Evolution dataset, even though we have known trait values at t_1 and t_{300}, the beginning and end of the simulated experiment.

References

Breiman, L. (2001). Random forests. Machine Learning 45(1), 5-32.

Kursa, M.B. and Rudnicki, W.R. (2010) Feature Selection with the Boruta Package. J. Stat. Softw. 36, 1–13

Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3),18-22.

Pichler, M. and Hartig, F. (2022) Machine Learning and Deep Learning – A review for Ecologists https://doi.org/10.48550/arXiv.2204.05023