Appendix 2

2025-11-18

So you wanna fit an integrated species distribution model? You've come to the right place!

NEED SOME SORT OF INTRO HERE.

Let's first load the libraries we'll need to set up, visualize, fit, and summarize our data.

```
# Load libraries ----
library(tidyverse)
library(sf)
library(nimble)

## To install SpFut.flexiSDM or check for updates, use the commented code below:
# remotes::install_github("rileymummah/SpFut.flexiSDM", build_vignettes = T)
library(SpFut.flexiSDM)
```

WHERE/HOW TO INTRODUCE RUNNING THIS ON AN HPC

To load data or fit a model, you must first create a .csv file that outlines the model specifications. We call ours `model-specs.csv`. Let's load the .csv and take a look at how we structured it.

```
mods <- read.csv("code/model-specs.csv")

head(mods, n=2)
```

```
##   number sp.code  model mem.nim mem.sum year.start year.end buffer sp.auto
## 1      1    RACA tau0.1   30000   40000       1994     2025  50000    TRUE
## 2      2    RACA   tau1   30000   40000       1994     2025  50000    TRUE
##   zero_mean tau coarse.grid cont.grid    covs.PO covs.inat
## 1      TRUE 0.1       FALSE     FALSE traveltime      <NA>
## 2      TRUE   1       FALSE     FALSE traveltime      <NA>
##                                   covs.lin covs.quad check.covs
## 1 sqrtarea_small, sqrtarea_medium, footprint TRI, tmin      FALSE
## 2 sqrtarea_small, sqrtarea_medium, footprint TRI, tmin      FALSE
##   covs.int.factor reference covs.int.cont Bpriordist Bpriorvar1 Bpriorvar2
## 1              NA        NA            NA      dnorm          0          1
## 2              NA        NA            NA      dnorm          0          1
##   filter.region spat.bal coordunc coordunc_na.rm block.rows block.cols
## 1          TRUE     TRUE     1000           TRUE          5          5
## 2          TRUE     TRUE     1000           TRUE          5          5
##   block.folds   iter thin region.sub lon.hi lon.lo lat.hi lat.lo project
## 1           3 100000    5       FALSE      NA      NA      NA      NA       NA
## 2           3 100000    5       FALSE      NA      NA      NA      NA       NA
##   exclude.dataset
## 1              NA
## 2              NA
```

INSERT TABLE DESCRIBING WHAT EACH COLUMN IS AND WHY IT'S INCLUDED IN THE .CSV

We have set up the scripts in the flexiSDM workflow so that a minimum number of parameters needs to be changed among them. To run this script in full (using our file arrangement), you only need to edit the following parameters:

```r
nums.do <- 2 # model number to run
block <- 'none' # block to run ('none', 1, 2, or 3)
local <- 1 # are you running the code locally (1) or on an HPC (0)?
a <- 1 # Fixed for indexing a vector below
```

Then we can use our model specifications to define a series of inputs for the setup script (found in `01-flexiSDM.R`).

```r
## Set up model variables
mods <- filter(mods, number %in% nums.do)

## Create all the combinations of model number and cross-validation block
tmp <- expand.grid(block.out = block, number = unique(mods$number))
mods <- full_join(mods, tmp, by = c("number"))
# CAN WE SIMPLIFY THIS SO THE CODE ONLY RUNS ONE MODEL NUMBER-BLOCK COMBO?
# THEN WE COULD REMOVE 'A' BELOW AND FIX IT AT 1

## Get variables for model from MVPv1.csv
number <- mods$number[a] # model number
sp.code <- mods$sp.code[a] # 4-digit species code
model <- mods$model[a] # model name
sp.auto <- mods$sp.auto[a] # include spatial autocorrelation?
coarse.grid <- mods$coarse.grid[a] # use a coarse grid?
cont.grid <- mods$cont.grid[a] # restrict to only a continuous grid?
year.start <- mods$year.start[a] # start year for data
year.end <- mods$year.end[a] # end year for data
buffer <- mods$buffer[a] # buffer size (km)
filter.region <- mods$filter.region[a]
spat.bal <- mods$spat.bal[a] # include spatial balancing?
coordunc <- mods$coordunc[a] # level of coordinate uncertainty to include (km)
coordunc_na.rm <- mods$coordunc_na.rm[a] # filter by coordinate uncertainty?
block.folds <- mods$block.folds[a] # how many groups of blocks?
block.rows <- mods$block.rows[a] # how many rows of blocks?
block.cols <- mods$block.cols[a] # how many columns of blocks?
block.out <- mods$block.out[a]

if (block.out == "none") {
  blockname <- "full" # rename the block to 'full' if not doing cross-validation
} else {blockname <- block.out} # otherwise, keep the block number
```

```r
# names of datasets to exclude (e.g., iNaturalist) - must match dataset naming
exclude.dataset <- unlist(str_split(mods$exclude.dataset[a], pattern = ", "))

## ICAR parameters
zero_mean <- mods$zero_mean[a] # zero mean assumption?
tau <- mods$tau[a] # precision (tau) can be a fixed value or a prior

## MCMC parameters
iter <- mods$iter[a] # number of MCMC iterations
thin <- mods$thin[a] # number to thin by
burnin <- floor(iter*0.75) # burnin to discard

## Change of region
region.sub <- mods$region.sub[a] # only estimate for a subregion?
lat.hi <- mods$lat.hi[a] # high value of latitude range
lat.lo <- mods$lat.lo[a] # low value of latitude range
lon.hi <- mods$lon.hi[a] # high value of longitude range
lon.lo <- mods$lon.lo[a] # low value of longitude range

## Future projections
project <- mods$project[a] # how many projections?
if (block.out != "none") {
  project <- 0 # no projections for cross-validation blocks
}

## Covariates
# list of PO covariates
covs.PO <- unlist(str_split(mods$covs.PO[a], pattern = ", "))
# list of iNaturalist covariates
covs.inat <- unlist(str_split(mods$covs.inat[a], pattern = ", "))
# list of linear covariates for state model
covs.lin <- unlist(str_split(mods$covs.lin[a], pattern = ", "))
# list of quadratic covariates for state model
covs.quad <- unlist(str_split(mods$covs.quad[a], pattern = ", "))

# SHOULD WE KEEP THIS IN? WE'RE NOT CURRENTLY USING IT BUT COULD??
# covs.int.factor <- unlist(str_split(mods$covs.int.factor[a], pattern = ", "))
# reference <- mods$reference[a]
# covs.int.cont <- unlist(str_split(mods$covs.int.cont[a], pattern = ", "))
check.covs <- mods$check.covs[a] # covariate selection to remove multicollinearity?

# Define the prior for the state model coefficients
Bpriordist <- mods$Bpriordist[a]
Bpriorvar1 <- mods$Bpriorvar1[a]
Bpriorvar2 <- mods$Bpriorvar2[a]
# WE COULD SIMPLIFY THIS TO LOOK LIKE TAU

process.intercept <- F # SHOULD RENAME TO state.intercept??
```