wolfAlps

Sarah Bauduin, Eliot McIntire

17 juin 2016

Overview

Demography and dispersal movement of the wolf in the Italian Alps. This is an R port of the original SELES model that was used in (Marucco and McIntire 2010).

Install SpaDES (R package)

For this and all subsequent code, you will need to be in R.

```
install.packages("devtools")
library(devtools)
install_github("PredictiveEcology/SpaDES@development")
install_github("PredictiveEcology/NetLogoR")
```

Download model

```
library(SpaDES)

moduleName <- "wolfAlps"

workingDirectory <- tempdir() # Change this to a place you would like to put the model, if desired

downloadModule(moduleName, data = TRUE, path = workingDirectory)
```

Usage

Below is the simple interactive section. If all that is desired is interactive work, then you need to go no further. For running non interactively, such as to get many replicates and probabilistic summaries of many runs, then continue on to subsequent sections.

```
moduleDir <- file.path(workingDirectory)

times <- list(start = 2008, end = 2008+14, timeunit = "year")

# These are the default parameters, none is being changed at the moment.
#parameters <- list(
  #.plotInitialTime = NA, .plotInterval = 1,
  #.saveInitialTime = 0, .saveInterval = 1
  # MeanNpups = 3.387, SdNpups = 1.210,
  # AdultMortalityRate  = 0.18, JuvMortalityRate = 0.449, DispMortRatePerMove = 0.0353,
```

```
  # MeanPackSize = 4.405, SdPackSize = 1.251,
  # EndDispersal = 0.98,
  # CellWidth = 1.25, MoveStep = 10.929,
  # sigma = 21.802,
  # MeanPixelQuality = 0.84, MinPixelQuality = 0.376, MinPackQuality = 89.288,
  # PackArea = 256, PhaseTransitionLower = 0.198,
  # run.tests = TRUE,
  # tImmigration = NA, locImmigrant = cbind(pxcor = NA, pycor = NA),
  # nGenes = 3, nAlleles = c(4,6,2)
#)

modules <- list("wolfAlps")

paths <- list(
  modulePath = moduleDir,
  cachePath = file.path(moduleDir, "outputR", "cache"),
  inputPath = file.path(moduleDir, "wolfAlps", "data"),
  outputPath =  file.path(moduleDir, "outputR")
)

# easier and more direct than via objects loading
inputs <- data.frame(file = c("wolves2008.asc", "packs2008.asc",
                              "CMR.asc", "HabitatSuitability.asc"))

# accept default parameters for now
wolfModuleStart <- simInit(times = times, #params = list(wolfAlps = parameters),
                 modules = modules, inputs = inputs, paths = paths)

wolfAlpsOutput <- spades(wolfModuleStart, progress = 14) # 14 updates to progress bar
```

Caching

Caching allows the user to save outputs without manually saving them. This can be rerun at later time and it will just reload results.

```
# if first time, must create a caching repo
# archivist::createLocalRepo(paths(wolfModuleStart)$cachePath)

wolfModuleStart <- simInit(times = times, #params = list(wolfAlps=parameters),
                 modules = modules, inputs = inputs, paths = paths)
wolfAlpsSim <- spades(sim = wolfModuleStart, cache = TRUE, replicate = 1) # , notOlderThan = Sys.time()
```

Run in parallel on one machine

Depending on your machine, this can speed things up substantially. The simulation uses about 600 MB of RAM. If the machine has enough RAM, then all threads can be used, with `makeCluster()`. For help on this, see experiment function help (i.e., `?experiment`). Below, we also use the cache = TRUE argument. This means that when this runs, the results will be cached. So, if the exact same call to `experiment` is made at a

later time, then it will just return the previous result. This can be convenient because it doesn't require the the user manually save objects; they are cached. However, for stochastic simulation, this may not always be desired. If no caching is desired, then either use `cache = FALSE` (the default, if cache argument is omitted), or add the argument `notOlderThan = Sys.time()`, will replace the older cached version with current run. Currently, this is set to run 11 replicates.

```
# for parallel version -- must begin the cache repository, see previous chunk
library(raster)
times <- list(start = 2008, end = 2008+14, timeunit = "year")
beginCluster()
# getCluster()
wolfModuleStart <- simInit(times = times, # params = list(wolfAlps=parameters),
                  modules = modules, inputs = inputs, paths = paths)
wolfAlpsSim2 <- experiment(wolfModuleStart, replicates = 11, cache = TRUE) # , notOlderThan = Sys.time()
endCluster()
```

Summaries

Taking the object created in the previous chunk, we can make summary figures. Here, we show just one such figure.

```
# look at probability of territory map
terrs <- lapply(wolfAlpsSim2, function(x) lapply(x$out_terr[(start(wolfAlpsSim2[[1]])+1):end(wolfAlpsSi
# create mean probability of territory map
TerritoryProb <- stack(list(Year2013=do.call(mean, lapply(terrs, function(x) x[[4]]>0)),
                        Year2018=do.call(mean, lapply(terrs, function(x) x[[9]]>0)),
                        Year2023=do.call(mean, lapply(terrs, function(x) x[[14]]>0))))
dev()
TerritoryProb[TerritoryProb>=0.3] <- 0.3
TerritoryProb <- stack(TerritoryProb)
Plot(TerritoryProb, cols = grey(9.9:0/10), legendRange = c(0, 0.3001))
```

Below here is for advanced users

Using ssh – Advanced user

If there is an ssh connection between machines, then PSOCK can be used. This means that the slave machines must be running an ssh server (usually on a linux machine), and there is a stored ssh key on each of those machines, authenticating the connection from the master.

```
library(parallel)
# make a cluster on 3 machines
machine1 <- PutMachine1Name
machine2 <- PutMachine2Name
machine3 <- "localhost"
NcoresOnEach <- 34 # can put this up to about 35
clNames <- c(rep(machine1, NcoresOnEach),
             rep(machine2, NcoresOnEach),
             rep(machine3, NcoresOnEach))
cl <- makeCluster(clNames, type = "PSOCK")
wolfModuleStart <- simInit(times = times, params = list(wolfAlps=parameters),
```

```
                  modules = modules, inputs = inputs, paths = paths)
wolfAlpsSim2 <- experiment(copy(wolfModuleStart), replicates = 100, cache = TRUE, .plotInitialTime = NA
                           cl = cl)

endCluster()
```

Events

Wolves age and some die. Juveniles become dispersers in packs where there are too many members. Subordinate adults replace missing alpha in packs. Alpha pairs reproduce. Dispersers walk away from the packs and either try to join a pack where there's only one alpha of the opposite sex or try to build a new territory if the suitability of the landscape is good enough. Some dispersers die if they haven't find a pack or new territory.

Plotting

Upper map: wolf territories (pack locations). Lower map: habitat suitability with wolf territories and individual locations.

Saving

Several information about wolves and packs is saved during the simulation. 9 outputs objects store them (out_terrSize, out_deaths, out_statPack, out_statInd, out_distDisp, out_statSim, out_terr, out_joinCreate, out_dispersers)

Data dependencies

Input data

Four raster maps are loaded prior running the model. These maps represent the wolves initial locations in 2008, the pack locations in 2008, CMR data about the wolves and a map of habitat suitability for the landscape.

Shiny

As with all SpaDES models, you can run a simple shiny app using `shine`.

```
shine(wolfModuleStart)
```

Output data

See ## Saving

SELES version

This model is a rewrite of a version originally written in SELES and used in the publication, (Marucco and McIntire 2010). This makes use of parallel package. The data files represent 100 replicate simulation outputs from the original SELES code.

```
selesDir <- file.path(moduleDir, "data")
outputMapDir <- tempdir()
unzip(file.path(selesDir, "outputSeles.zip"), exdir = outputMapDir)

library(parallel)
cl <- makeCluster(detectCores()-2)
clusterEvalQ(cl, library(raster))

sYear2023files <- dir(outputMapDir, pattern = "Packs_.*_2023_", recursive = TRUE, full.names = TRUE)
sYear2023 <- clusterApplyLB(cl, sYear2023files, function(x) {
  out <- raster(x)
  out[out>0] <- 1
  out
  })

sYear2018files <- dir(outputMapDir, pattern = "Packs_.*_2018_", recursive = TRUE, full.names = TRUE)
sYear2018 <- clusterApplyLB(cl, sYear2018files, function(x) {
  out <- raster(x)
  out[out>0] <- 1
  out})

sYear2013files <- dir(outputMapDir, pattern = "Packs_.*_2013_", recursive = TRUE, full.names = TRUE)
sYear2013 <- clusterApplyLB(cl, sYear2013files, function(x) {
  out <- raster(x)
  out[out>0] <- 1
  out})


sTerritoryProb <- stack(list(Year2013=do.call(mean, sYear2013),
                       Year2018=do.call(mean, sYear2018),
                       Year2023=do.call(mean, sYear2023)))
#sTerritoryProb[sTerritoryProb>=0.3] <- 0.3
sTerritoryProb <- stack(sTerritoryProb)
Plot(sTerritoryProb, cols = grey(9.9:0/10), new=TRUE)

TerritoryProb2 <- stack(sTerritoryProb)
TerritoryProb2Stack <- stack(lapply(1:3, function(x) {
  TerritoryProb2[[x]][] <-  TerritoryProb[[x]][]
  TerritoryProb2[[x]]}))
diffProbList <- lapply(1:3, function(x) 10*round(TerritoryProb2Stack[[x]] - sTerritoryProb[[x]],1))
names(diffProbList) <- paste0("Year", c(2013, 2018, 2023))
diffProb <- stack(lapply(diffProbList, function(x) x/10))
diffProb[diffProb==0] <- NA
diffProb <- stack(diffProb)
Plot(diffProb, new=TRUE, zero.color = "white", cols = divergentColors("red", "green", -5, 5, 0, "white"
```

Literature

Marucco, F., and E. J. B. McIntire. 2010. Predicting spatio-temporal recolonization of large carnivore populations and livestock depredation risk: Wolves in the Italian Alps. Journal of Applied Ecology 47:789–798.