Appendix S2

Maëlis Kervellec

2023-09-13

```r
require(nimble)
```

```
## Loading required package: nimble

## nimble version 0.13.1 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit https://R-nimble.org.
##
## Note for advanced users who have written their own MCMC samplers:
##   As of version 0.13.0, NIMBLE's protocol for handling posterior
##   predictive nodes has changed in a way that could affect user-defined
##   samplers in some situations. Please see Section 15.5.1 of the User Manual.

##
## Attaching package: 'nimble'

## The following object is masked from 'package:stats':
##
##     simulate
```

1. Dynamic occupancy model

```r
mackenzie.model <- nimbleCode({

  # Priors: ecological parameters
  psi1 ~ dunif(0, 1)  # initial occupancy probability
  gamma ~ dunif(0, 1) # colonization probability
  phi ~ dunif(0, 1)   # extinction probability
  p ~ dunif(0, 1)     # detection probability

  # ecological submodel
  for (i in 1:nsites){
    z[i,1] ~ dbern(psi1)

    for (t in 2:nseasons){
      muZ[i,t]<- z[i,t-1] * phi + (1 - z[i,t-1]) * gamma
      z[i,t] ~ dbern(muZ[i,t])
    }
```

```
  }

  # observation model
  for (i in 1:nsites){
    for (t in 1:nseasons){
      y[i,t] ~ dbinom(size = nsurveys, prob = z[i,t] * p)
    }
  }
})
```

2. Spatial dynamic occupancy model

```
## Define distance -------------------------------------------------------------
d <- AHMbook::e2dist(xy,xy) # euclidean distance between all pairs of sites
distSq <- (d^2)/sd(d^2)

## Model -----------------------------------------------------------------------
euclidean.model <- nimble::nimbleCode({

  # Priors: ecological parameter
  psi1 ~ dunif(0, 1)   # first session occupancy probability
  gamma0 ~ dunif(0, 1) # baseline colonization probability
  sigma ~ dgamma(shape = 1, rate = 1) # scale parameter
  phi ~ dunif(0, 1)   # extinction probability
  p ~ dunif(0, 1)     # detection probability

  # ecological submodel
  for (i in 1:nsites){
    z[i,1] ~ dbern(psi1)

    for(t in 2:nseasons) {
      for(n in 1:nsites) {
        # pairwise colonization probability
        gammaDistPairs[i,n,t-1] <- 1 - gamma0 * exp(-distSq[i,n] / (2 * sigma^2)) * z[n,t-1]
      }
      # colonization probability
      gamma[i,t-1] <- 1 - prod(gammaDistPairs[i,1:nsites,t-1])
      ## Pr(z=1|z=1) = 1 - Pr(extinction)*Pr(not colonized) = 1 - ((1-phi)*(1-gamma))
      muz[i,t-1] <- gamma[i,t-1] * (1 - z[i,t-1]) + phi * z[i,t-1]
      z[i,t] ~ dbern(muz[i,t-1])
    }
  }
  # observation model
  for (i in 1:nsites){
    for (t in 1:nseasons){
      y[i,t] ~ dbinom(size = nsurveys, prob = z[i,t] * p)
    }
  }
})
```

3. Spatial dynamic occupancy model accomodated with Least Cost Path distance

```r
## Define distance ------------------------------------------------------------
leastcostpath <- function(alpha){
  # Compute least cost path between a set of points given a resistance parameter
  # alpha and a resistance surface rcov

  # alpha is a scalar
  # rcov is a RasterLayer
  # xy is a matrix of dims nsites x 2

  # obtain resistance surface
  cost <- exp(alpha * rcov)
  # compute conductances among neighbours
  # probability of transition from one cell to another
  tr <- gdistance::transition(x = cost,
                              transitionFunction = function(x) 1/mean(x),
                              directions = 8) # class TransitionLayer
  # adjust diag.conductances
  trCorrC <- gdistance::geoCorrection(x = tr,
                                      type = "c",
                                      multpl = FALSE,
                                      scl = FALSE) # class TransitionLayer
  # compute ecological distance
  d <- gdistance::costDistance(x = trCorrC,
                               fromCoords = xy,
                               toCoords = xy) # class Matrix
  d^2
}

# Create the nimble function to compute distance between sites
LCP <- nimbleRcall(function(alpha = double(0)){},
                   Rfun = 'leastcostpath',
                   returnType = double(2))

## Model ----------------------------------------------------------------------
lcp.model <- nimble::nimbleCode({

  # Priors: ecological parameter
  psi1 ~ dunif(0, 1)   # first session occupancy probability
  gamma0 ~ dunif(0, 1) # baseline colonization probability
  alpha2 ~ dunif(-5, 5) # resistance parameter
  sigma ~ dgamma(shape = 1, rate = 1) # scale parameter
  phi ~ dunif(0, 1)   # extinction probability
  p ~ dunif(0, 1)     # detection probability

  # computation of distances for a given resistance
  distSq[1:nsites,1:nsites] <- LCP(alpha2)
  distSqreduced[1:nsites,1:nsites] <- distSq[1:nsites,1:nsites] / sd(distSq[1:nsites,1:nsites])
  # reduced to help convergence
```

```
  # ecological submodel
  for (i in 1:nsites){
    z[i,1] ~ dbern(psi1)

    for(t in 2:nseasons) {
      for(n in 1:nsites) {
        # pairwise colonization probability
        gammaDistPairs[i,n,t-1] <- 1 - gamma0 * exp(-distSqreduced[i,n] / (2 * sigma^2)) * z[n,t-1]
      }
      # colonization probability
      gamma[i,t-1] <- 1 - prod(gammaDistPairs[i,1:nsites,t-1])
      muz[i,t-1] <- gamma[i,t-1] * (1 - z[i,t-1]) + phi * z[i,t-1]
      z[i,t] ~ dbern(muz[i,t-1])
    }
  }
  # observation model
  for (i in 1:nsites){
    for (t in 1:nseasons){
      y[i,t] ~ dbinom(size = nsurveys, prob = z[i,t] * p)
    }
  }
})
```

3. Spatial dynamic occupancy model accomodated with commute distance

```
## Define distance -------------------------------------------------------------
circuitdistance <- function(alpha){
  # alpha is a scalar
  # rcov is a RasterLayer
  # xy is a matrix of dims nsites x 2
  # obtain resistance surface
  cost <- exp(alpha * rcov)
  # compute conductances among neighbours
  # probability of transition from one cell to another
  tr <- gdistance::transition(x = cost,
                              transitionFunction = function(x) 1/mean(x),
                              directions = 8) # class TransitionLayer
  # adjust diag.conductances
  trCorrC <- gdistance::geoCorrection(x = tr,
                                      type = "r", # for randomwalk
                                      scl = TRUE) # class TransitionLayer

  #Circuit theory = random walk with theta = 0
  circuitDist <- gdistance::commuteDistance(x = trCorrC, coords = xy) %>%
    as.matrix()
  dist <- circuitDist
  return(dist^2)
}
```

```r
CircuitDistance <- nimbleRcall(function(alpha = double(0)){},
                              Rfun = 'circuitdistance',
                              returnType = double(2))

## Model -----------------------------------------------------------------------
circuit.model <- nimble::nimbleCode({

  # Priors: ecological parameter
  psi1 ~ dunif(0, 1)   # first session occupancy probability
  gamma0 ~ dunif(0, 1) # baseline colonization probability
  alpha2 ~ dunif(-5, 5) # resistance parameter
  sigma ~ dgamma(shape = 1, rate = 1) # scale parameter
  phi ~ dunif(0, 1)   # extinction probability
  p ~ dunif(0, 1)     # detection probability

  # computation of distances for a given resistance
  distSq[1:nsites,1:nsites] <- CircuitDistance(alpha2)
  distSqreduced[1:nsites,1:nsites] <- distSq[1:nsites,1:nsites] / sd(distSq[1:nsites,1:nsites])

  # ecological submodel
  for (i in 1:nsites){
    z[i,1] ~ dbern(psi1)

    for(t in 2:nseasons) {
      for(n in 1:nsites) {
        # pairwise colonization probability
        gammaDistPairs[i,n,t-1] <- 1 - gamma0 * exp(-distSqreduced[i,n] / (2 * sigma^2)) * z[n,t-1]
      }
      # colonization probability
      gamma[i,t-1] <- 1 - prod(gammaDistPairs[i,1:nsites,t-1])
      muz[i,t-1] <- gamma[i,t-1] * (1 - z[i,t-1]) + phi * z[i,t-1]
      z[i,t] ~ dbern(muz[i,t-1])
    }
  }
  # observation model
  for (i in 1:nsites){
    for (t in 1:nseasons){
      y[i,t] ~ dbinom(size = nsurveys, prob = z[i,t] * p)
    }
  }
})
```