demoniche

Hedvig Nenzén

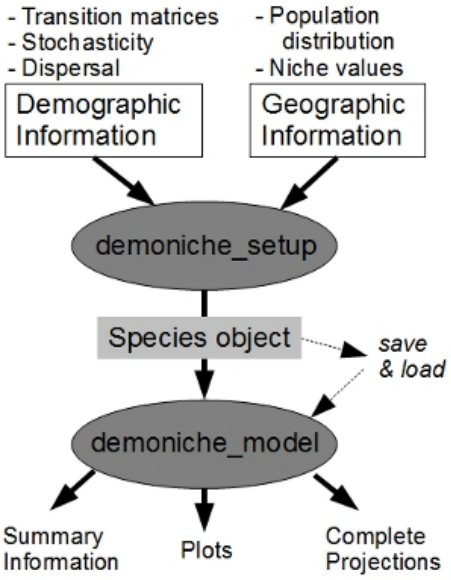
August 16, 2011

1 Introduction

`demoniche` is a freely available R-package to simulate stochastic population growth for various subpopulations of a species. Demographic models projects population sizes with various transition matrices that represent demographic impacts on species growth. The Demographic modelling is linked to a time series of geographically distributed 'Niche values' that also affect species growth. The demoniche model offers flexible options for stochasticity, density dependence and dispersal. With the demoniche package it is possible to investigate population sizes, extinction probabilities and range shift of a species under the influence of scenarios of environmental and human impacts.

The main steps to running a model are as follows:

- Create an object with `demoniche_setup` function that contains all the information about the species that is being modelled (demographical, geographical)
- Run the `demoniche` function on the species object
- Analyse the results



2 How to use demoniche

2.1 Install the package

To install the demoniche package please type install.packages("demoniche", repos="http://R-Forge.R-project.org") in R. Or go to the webpage http://demoniche.r-forge.r-project.org/ and follow the directions.

The demoniche package depends on functions and data from three other packages; popbio, lattice and sp that are available from CRAN. They should install automatically, please do it manually if not.

Set the working directory, and then load the package.

```
> library(demoniche)
```

The code in this manual is also provided in the demoniche_manual.R script in the /doc folder where the package is saved, normally this is at C:\Program Files\R\R-2.13.0\library, or a similar location.

2.2 Loading data supplied with package

We load the example data file supplied in the package. The object is called Hmontanta and contains demographic and geographic data about Mountain Goldenheater *Hudsonia montana* (Gross et al. 1998). We can inspect the object with str(). We find that Hmontana is a list, so we can look at parts of the list using $. This object contains all the information needed about the species to carry out modelling.

```
> data(Hmontana)
```

```
> str(Hmontana)
```

```
List of 26
 $ Orig_Populations          :'data.frame':        34 obs. of  4 variables:
  ..$ PatchID        : int [1:34] 8000 8001 8002 8003 8004 8005 8006 8007 8008 8009 ...
  ..$ XCOORD         : int [1:34] 2 3 6 7 9 11 12 17 18 19 ...
  ..$ YCOORD         : int [1:34] 29 29 29 29 29 29 29 29 29 29 ...
  ..$ area_population: int [1:34] 2 2 2 2 2 2 2 2 2 2 ...
 $ fraction_SDD              : num 0.5
 $ dispersal_probabilities   : num [1:900, 1:900] 0.0 8.7e-08 0.0 0.0 0.0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:900] "5000" "5001" "5002" "5003" ...
  .. ..$ : chr [1:900] "5000" "5001" "5002" "5003" ...
 $ dist_latlong              : num [1:900, 1:900] 0 1 2 3 4 5 6 7 8 9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:900] "5000" "5001" "5002" "5003" ...
  .. ..$ : chr [1:900] "5000" "5001" "5002" "5003" ...
 $ neigh_index               : num [1:2] 1 1.4
 $ Niche_ID                  :'data.frame':        900 obs. of  4 variables:
  ..$ Niche_ID    : int [1:900] 5000 5001 5002 5003 5004 5005 5006 5007 5008 5009 ...
  ..$ X           : int [1:900] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ Y           : int [1:900] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ PopulationID: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
 $ Niche_values              :'data.frame':        900 obs. of  9 variables:
  ..$ period2000: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2010: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2020: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2030: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
```

```
  ..$ period2040: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2050: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2060: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2070: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ period2080: num [1:900] 0 0 0 0 0 0 0 0 0 0 ...
 $ years_projections         : chr [1:9] "period2000" "period2010" "period2020" "period2030" ...
 $ matrices                  : num [1:36, 1:5] 0.4995 0.0004 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:5] "Reference_matrix" "Mx1" "Mx2" "Mx3" ...
 $ matrices_var              : num [1:36, 1] 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ..
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "sd"
 $ prob_scenario             : num [1:2] 0.5 0.5
 $ noise                     : num 0.95
 $ stages                    : chr [1:6] "seed" "seedlings" "tiny" "small" ...
 $ proportion_initial        : num [1:6] 0.98181 0.000691 0.006908 0.003684 0.005756 ...
 $ density_individuals       : num [1:34] 20000 20000 20000 20000 20000 20000 20000 20000 20000 2
 $ fraction_LDD              : num 0.05
 $ no_yrs                    : num 10
 $ K                         : num 100
 $ populationmax_all         : num [1:900] 4e+06 4e+06 4e+06 4e+06 4e+06 4e+06 4e+06 4e+06 4e+06
 $ n0_all                    : num [1:900, 1:6] 0 0 0 0 0 0 0 0 0 0 ...
 $ list_names_matrices       :List of 5
  ..$ : chr "Reference_matrix"
  ..$ : chr "Mx1"
  ..$ : chr "Mx2"
  ..$ : chr "Mx3"
  ..$ : chr "Mx4"
 $ sumweight                 : num [1:6] 0 1 1 1 1 1
 $ transition_affected_env   : int [1:24] 1 2 9 13 14 15 16 19 20 21 ...
 $ transition_affected_niche : num [1:2] 1 3
 $ transition_affected_demogr: int [1:24] 1 2 9 13 14 15 16 19 20 21 ...
 $ env_stochas_type          : chr "normal"
NULL
```

```
> Hmontana$env_stochas_type
```

```
[1] "normal"
```

2.3 Modelling

we use the modelling function `demoniche_model(modelname, Niche, Dispersal, repetitions, foldername)` to carry out the demographic modelling, and specifying the `Hmontana` list of species information as the species information we want to use. As arguments to the function the user also needs to specify if you want to run simulations with the effects of the Niche values (TRUE or FALSE) and if you want to allow long-distance dispersal (TRUE or FALSE). You also need to specify how many repetitions you want to carry out (for stochastic simulations the number should be over 1000), and a name for the folder where the simulations will be stored. But all these simulations are carried out with the same species information.

The `demoniche_model` function runs two internal functions, `demoniche_population` that carries out demographic modelling, and `demoniche_dispersal` which calculates the dispersal if selected.

When we run the `demoniche_model` function messages are printed on the screen, to let us know how the simulations are going.

```
> noCC_nodispersal <- demoniche_model(modelname = "Hmontana",
+     Niche = FALSE, Dispersal = FALSE, repetitions = 2,
+     foldername = "noCC_nodispersal")

[1] Starting projections for repetition: 1
[1] Projecting for scenario/matrix: Reference_matrix
[1] Projecting for scenario/matrix: Mx1
[1] Projecting for scenario/matrix: Mx2
[1] Projecting for scenario/matrix: Mx3
[1] Projecting for scenario/matrix: Mx4
[1] Starting projections for repetition: 2
[1] Projecting for scenario/matrix: Reference_matrix
[1] Projecting for scenario/matrix: Mx1
[1] Projecting for scenario/matrix: Mx2
[1] Projecting for scenario/matrix: Mx3
[1] Projecting for scenario/matrix: Mx4
[1] Calculating summary values
[1]     All repetitions completed!
```

To change any of the parameters, we simply change the arguments in the function, and the foldername, to change where the objects are saved. Here we have chosen to include effects of Niche values but no dispersal.

```
> CC_nodispersal <- demoniche_model(modelname = "Hmontana",
+     Niche = TRUE, Dispersal = FALSE, repetitions = 2,
+     foldername = "CC_nodispersal")

[1] Starting projections for repetition: 1
[1] Projecting for scenario/matrix: Reference_matrix
[1] Projecting for scenario/matrix: Mx1
[1] Projecting for scenario/matrix: Mx2
[1] Projecting for scenario/matrix: Mx3
[1] Projecting for scenario/matrix: Mx4
[1] Starting projections for repetition: 2
[1] Projecting for scenario/matrix: Reference_matrix
[1] Projecting for scenario/matrix: Mx1
[1] Projecting for scenario/matrix: Mx2
[1] Projecting for scenario/matrix: Mx3
[1] Projecting for scenario/matrix: Mx4
[1] Calculating summary values
[1]     All repetitions completed!
```

2.4 Analyse data

From running the `demoniche_model` function we get these outputs:

- Summary statistics from all the repetitions in the workspace
- Plots of population sizes and patch occupancy
- Simulation results in csv file
- Population data saved in folder, as R object

- Eigen analysis saved in folder, as R object

First, the output from the `demoniche_model` function itself is an array containing the population sizes at each time step (mean, Expected Minimum Abundance (EMA), and standard deviation of mean population sizes), calculated from all repetitions. The third dimension of the array is the different treatment matrices.

Complete yearly population stage distributions from each repetition and each population are saved in the folder with the specified name. These data can be visually or statistically analysed, or exported to other formats for further analysis of transient dynamics. The function also creates line graphs of the EMA of each transition matrix treatment scenario, and maps of the occupancy throughout the modelled area, at each time step (Fig. 2)

```
> dim(noCC_nodispersal)

[1] 90  3  5

> dimnames(noCC_nodispersal)

[[1]]
 [1] "year1"  "year2"  "year3"  "year4"  "year5"  "year6"
 [7] "year7"  "year8"  "year9"  "year10" "year11" "year12"
[13] "year13" "year14" "year15" "year16" "year17" "year18"
[19] "year19" "year20" "year21" "year22" "year23" "year24"
[25] "year25" "year26" "year27" "year28" "year29" "year30"
[31] "year31" "year32" "year33" "year34" "year35" "year36"
[37] "year37" "year38" "year39" "year40" "year41" "year42"
[43] "year43" "year44" "year45" "year46" "year47" "year48"
[49] "year49" "year50" "year51" "year52" "year53" "year54"
[55] "year55" "year56" "year57" "year58" "year59" "year60"
[61] "year61" "year62" "year63" "year64" "year65" "year66"
[67] "year67" "year68" "year69" "year70" "year71" "year72"
[73] "year73" "year74" "year75" "year76" "year77" "year78"
[79] "year79" "year80" "year81" "year82" "year83" "year84"
[85] "year85" "year86" "year87" "year88" "year89" "year90"

[[2]]
[1] "Meanpop" "EMA"     "SD"

[[3]]
[1] "Reference_matrix" "Mx1"              "Mx2"
[4] "Mx3"              "Mx4"
```

We can also look at parts of the object itself, here the Expected Mimimum Abundance from simulations with the treatment matrix Mx4:

```
> noCC_nodispersal[, "EMA", "Mx4"]

  year1   year2   year3   year4   year5   year6   year7
  27776   29171   30789   30706   32501   33477   34662
  year8   year9  year10  year11  year12  year13  year14
  38303   38557   38731   38976   45100   45566   47110
 year15  year16  year17  year18  year19  year20  year21
  50261   50404   52212   54979   61295   64958   68928
 year22  year23  year24  year25  year26  year27  year28
  69752   71213   73609   81034   85863   83631   93555
 year29  year30  year31  year32  year33  year34  year35
```

```
 95521   95436  101047  101351  111181  112648  117507
year36  year37  year38  year39  year40  year41  year42
113416  121868  128331  131647  144265  143254  145042
year43  year44  year45  year46  year47  year48  year49
153912  155117  163953  168880  191881  198459  204516
year50  year51  year52  year53  year54  year55  year56
216063  240954  239258  248451  269012  262845  289055
year57  year58  year59  year60  year61  year62  year63
310388  326086  332364  340046  366190  400032  410442
year64  year65  year66  year67  year68  year69  year70
459126  456262  481023  522538  534994  535199  535156
year71  year72  year73  year74  year75  year76  year77
561997  573288  600731  656799  677274  699460  738832
year78  year79  year80  year81  year82  year83  year84
750322  783438  766775  847893  901188  908232  952568
year85  year86  year87  year88  year89  year90
932625 1038090 1111415 1172035 1195069 1286106
```

The values obtained in other runs will be different as their population growth is stochastic.

We can plot the population sizes. Figure 2 is produced by the following code:

```
> barplot(cbind(noCC_nodispersal[90, 2, ], CC_nodispersal[90,
+     2, ]), beside = TRUE, legend.text = Hmontana$list_names_matrices,
+     names.arg = c("no Niche values", "with Niche values"))
```

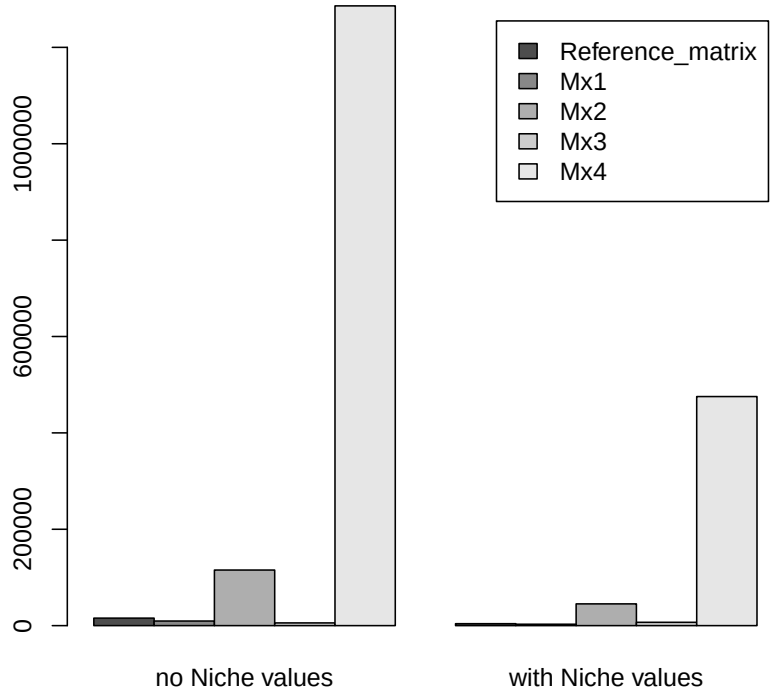


Figure 2: Population size (number of individuals) for simulations under each treatment matrix scenario of human land use, in the last year of simulation. With climate change there was a lower predicted population size

Apart from the summary object, the results from each simulation are saved in the folder with the specified name, in the working directory. We can look at what they are called:

```
> list.files(path = "noCC_nodispersal")
```

```
 [1] "eigen_results.rda"
 [2] "EMA_Mx1.jpeg"
 [3] "EMA_Mx2.jpeg"
 [4] "EMA_Mx3.jpeg"
 [5] "EMA_Mx4.jpeg"
 [6] "EMA_Reference_matrix.jpeg"
 [7] "map_Mx1.jpeg"
 [8] "map_Mx2.jpeg"
 [9] "map_Mx3.jpeg"
[10] "map_Mx4.jpeg"
[11] "map_Reference_matrix.jpeg"
[12] "metapop_results.rda"
[13] "population_sizes.rda"
[14] "Projection_rep1_Mx1.rda"
[15] "Projection_rep1_Mx2.rda"
[16] "Projection_rep1_Mx3.rda"
[17] "Projection_rep1_Mx4.rda"
[18] "Projection_rep1_Reference_matrix.rda"
[19] "Projection_rep2_Mx1.rda"
[20] "Projection_rep2_Mx2.rda"
[21] "Projection_rep2_Mx3.rda"
[22] "Projection_rep2_Mx4.rda"
[23] "Projection_rep2_Reference_matrix.rda"
[24] "simulation_results.csv"
[25] "simulation_results.rda"
```

In the folder there are various .jpgs that show the EMA through time for all repetitions, and the patch occupancy at each time period, for the last repetition. These are intended for a quick visualization of results. The original data can be accessed and plotted as the user wants. Other results are saved in `.csv` which can be opened in for example excel, such as the `simulation_results.csv`. The rows are the information for each matrix. The columns contain information about the projects: initial total population area, initial population, percentage of populations that went extinct during the simulations, the lambda (λ, first eigenvalue) of the treatment matrix (not the reference matrix, function from `popbio` package), the stochastic lambda from the two matrices (50.000 time intervals, function from `popbio` package). *Some of these reults might not be working correctly.*

Other data, such as the complete occupancy for each population and each repetition, for all stages, are saved in other formats. The `.rda` format can be loaded with the command `load()` into R. This data can then be used for plotting, further analyzed, etc.

The user can also load the object `eigen_results_yourmatrixname.rda` for each transition matrix. It is a list with an entry for each matrix, which is made up of the results from the function `eigen.analysis` from the `popbio` package. The list contains the eigenvalue of the matrix (λ), the stable stage distribution, the sensitivities and elasticities, the reproductive value of each life stage, and the damping ratio. For explanation of these values please see `popbio` package helpfiles, Morris and Doak, 2002 and other publications listed below.

```
> load("noCC_nodispersal/eigen_results.rda")
```

```
[1] "eigen_results"
```

```
> str(eigen_results)
```

```
List of 5
 $ Reference_matrix:List of 7
  ..$ lambda1      : num 0.959
  ..$ stable.stage : num [1:6] 0.98142 0.00075 0.00398 0.00342 0.00476 ...
  ..$ sensitivities: num [1:6, 1:6] 0.0113 12.9282 0 0 0 ...
  ..$ elasticities : num [1:6, 1:6] 0.00587 0.00539 0 0 0 ...
  ..$ repro.value  : num [1:6] 1 1148 2305 4340 5552 ...
  ..$ damping.ratio: num 1.4
  ..$ LTRE         : num [1:6, 1:6] 0 0 0 0 0 0 0 0 0 0 ...
 $ Mx1             :List of 7
  ..$ lambda1      : num 0.959
  ..$ stable.stage : num [1:6] 0.974 0.0008 0.0075 0.00653 0.00811 ...
  ..$ sensitivities: num [1:6, 1:6] 0.00964 11.07891 0 0 0 ...
  ..$ elasticities : num [1:6, 1:6] 0.00502 0.00462 0 0 0 ...
  ..$ repro.value  : num [1:6] 1 1150 2311 4287 4661 ...
  ..$ damping.ratio: num 1.44
  ..$ LTRE         : num [1:6, 1:6] 0 0 0 0 0 0 0 0 0 0 ...
 $ Mx2             :List of 7
  ..$ lambda1      : num 1.01
  ..$ stable.stage : num [1:6] 0.987575 0.000812 0.001046 0.000606 0.000291 ...
  ..$ sensitivities: num [1:6, 1:6] 0.00926 11.8108 0 0 0 ...
  ..$ elasticities : num [1:6, 1:6] 0.00458 0.00468 0 0 0 ...
  ..$ repro.value  : num [1:6] 1 1276 2699 4696 7653 ...
  ..$ damping.ratio: num 1.29
  ..$ LTRE         : num [1:6, 1:6] 0 0 0 0 0 0 0 0 0 0 ...
 $ Mx3             :List of 7
  ..$ lambda1      : num 0.845
  ..$ stable.stage : num [1:6] 0.979559 0.000598 0.006777 0.005622 0.004388 ...
  ..$ sensitivities: num [1:6, 1:6] 0.00389 4.48501 0 0 0 ...
  ..$ elasticities : num [1:6, 1:6] 0.0023 0.00159 0 0 0 ...
  ..$ repro.value  : num [1:6] 1 1153 2144 7201 22964 ...
  ..$ damping.ratio: num 1.26
  ..$ LTRE         : num [1:6, 1:6] 0 0 0 0 0 0 0 0 0 0 ...
 $ Mx4             :List of 7
  ..$ lambda1      : num 1.02
  ..$ stable.stage : num [1:6] 0.986585 0.000589 0.00127 0.000474 0.002607 ...
  ..$ sensitivities: num [1:6, 1:6] 0.0152 26.2875 0 0 0 ...
  ..$ elasticities : num [1:6, 1:6] 0.00746 0.00774 0 0 0 ...
  ..$ repro.value  : num [1:6] 1 1729 3522 4523 4990 ...
  ..$ damping.ratio: num 1.36
  ..$ LTRE         : num [1:6, 1:6] 0 0 0 0 0 0 0 0 0 0 ...
NULL
```

We can use the `popbio` package function `image2` to plot the Sensitivity matrix (Figure 3). We can see that the stage with the highest sensitivity is the stage from seed to seedling (12.928).

```
$mar
[1] 1 3 3 1

$xpd
[1] TRUE

NULL
```

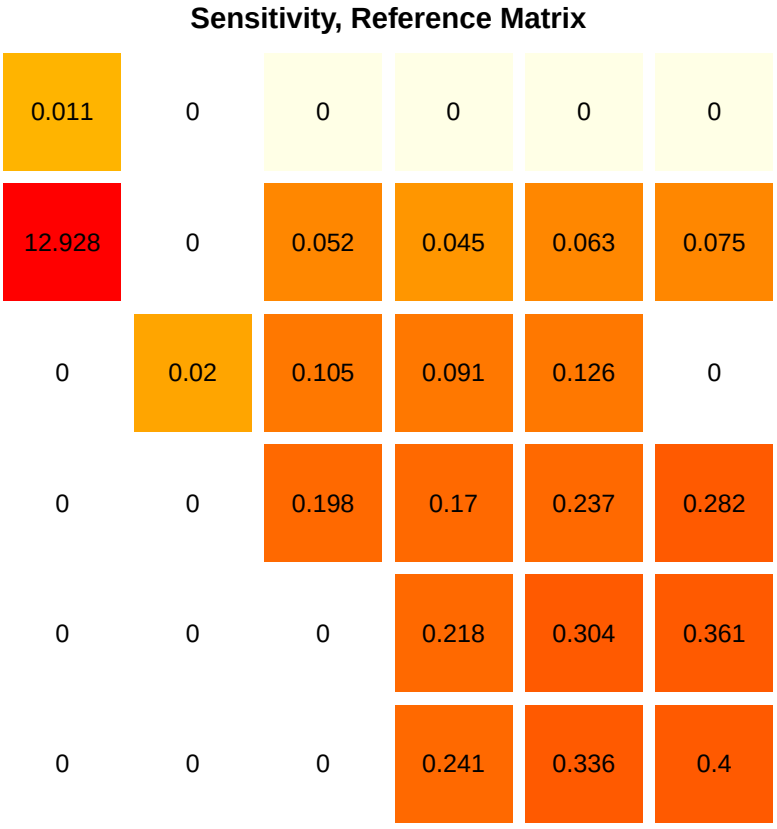


Figure 3: Sensitivities of the Reference Matrix for Mountain Goldenheater *Hudsonia montana*.

3 Setting up our own data

The `demoniche` model, and demographic modelling in general is 'data-hungry' and requires both detailed geographic and demographic information for the modelled species. For further discussion about how to obtain data, please see Morris and Doak (2002). Overview of required information:

Geographic information:

- Information about occupied populations, with at least two sub-populations. Each patch must have coordinates, a (numeric) identifier, and a size (size can be the same for all patches if the model is grid-based).
- A grid of future conditions (Niche values), from 0-1 or 0-1000. They can be outputs from niche modelling (Species Distribution Modelling), scenarios derived from General Circulation Models (GCM), predictions of land use-changes, habitat cover types, topology, etc. They can also be any other values thought to influence the species growth (they can also be zeros if the surrounding environment is completely unsuitable for the species).

Demographic information:

- Age(Leslie)- or stage(Lefkovitch)-based transition matrix
- Optional variance matrix for each value in the transition matrix.
- Various 'treatment' matrices that represent a certain environmental effect (or threat). At each year in the simulation, either the mean matrix will be chosen, or the 'treatment' matrix with a user-definied probability.
- The initial proportion of individuals in each stage.
- The fraction that seeds that undergo short and long distance dispersal.
- The maximum population size for each patch, or a multiple above which the population cannot grow. The multiple can either be a value for each patch or one value for all patches.
- Which stages in the transition matrix which are affected by environmental and demographic stochasticity.

As an example of how we would load and model with our own data, we here re-create our example file `Hmontana` (Gross et al. 1998). We show how we could load our own information about a species, step by step. When we run `demoniche_setup` function on this information, we create the object `Hmontana`. This example can also be read directly from the R-script `demoniche_manual.R` which is located in the 'doc' folder in the package.

The information can either be written into the workspace, or read from for example `.csv` files. They can either be defined as objects, or defined directly in the `demoniche_setup(...)` function. These objects can have any name, and in this example we have named them with the suffix `_mine` for clarity. Then, when running the setup function we specify which object corresponds to which argument in the function. For example, instead of `no_yrs_mine <- 10` and then defining the object in the setup function: `demoniche_setup(... no_years = no_years_mine)`, we can simple write the value `demoniche_setup(... no_years = 10)`. This is easier for short values.

Each argument needs either an already defined object or a value, except those that already have a default.

3.1 Geographic information

Original population distribution. Should be one file with ID of patches (must be numeric), coordinates of original locations, area of the population (in same unit as density data). If all populations have the same size (grid-based) the area will be the same for all populations. The data should be in lat/long format.

```
> Populations_mine <- read.table(file = "Hudsonia_Populations_grids.csv",
+     sep = ",", header = TRUE)
> head(Populations_mine)
```

```
  patchID  X  Y area
1    8000  2 29    2
2    8001  3 29    2
3    8002  6 29    2
4    8003  7 29    2
5    8004  9 29    2
6    8005 11 29    2
```

Niche data. This can be a regular grid comprising the locations of the populations. In this case we are using predicted proabilities of presence under climate change (from a GLM).

```
> Nichemap_mine <- read.table(file = "Hudsonia_SDMmodelling.csv",
+     sep = ",", header = TRUE)
> tail(Nichemap_mine)
```

```
    gridID  X  Y period2000 period2010 period2020
895   5894 25 30        0.9       0.87          0
896   5895 26 30        0.9       0.87          0
897   5896 27 30        0.9       0.87          0
898   5897 28 30        0.9       0.87          0
899   5898 29 30        0.9       0.87          0
900   5899 30 30        0.9       0.87          0
    period2030 period2040 period2050 period2060 period2070
895       0.73       0.67          0          0          0
896       0.73       0.67          0          0          0
897       0.73       0.67          0          0          0
898       0.73       0.67          0          0          0
899       0.73       0.67          0          0          0
900       0.73       0.67          0          0          0
    period2080
895          0
896          0
897          0
898          0
899          0
900          0
```

With the `lattice` package we can plot the Niche values and see how the most suitable areas move towards the south, Figure 3.1.

```
period2000 + period2010 + period2020 + period2030 + period2040 +
    period2050 + period2060 + period2070 + period2080 ~ X + Y
```

```
> print(levelplot(niche_formulas, Nichemap_mine,
+     col.regions = rev(heat.colors(100)), main = "Niche Values"))
```

3.2 Demographic information

A matrix/data frame of transtion matrices for different scenarios, one column per matrix. The dimension should be a multiple of the number of stages. The first matrix should be the 'reference' matrix. The model can also run with a single matrix.

Load required data from the popbio package:

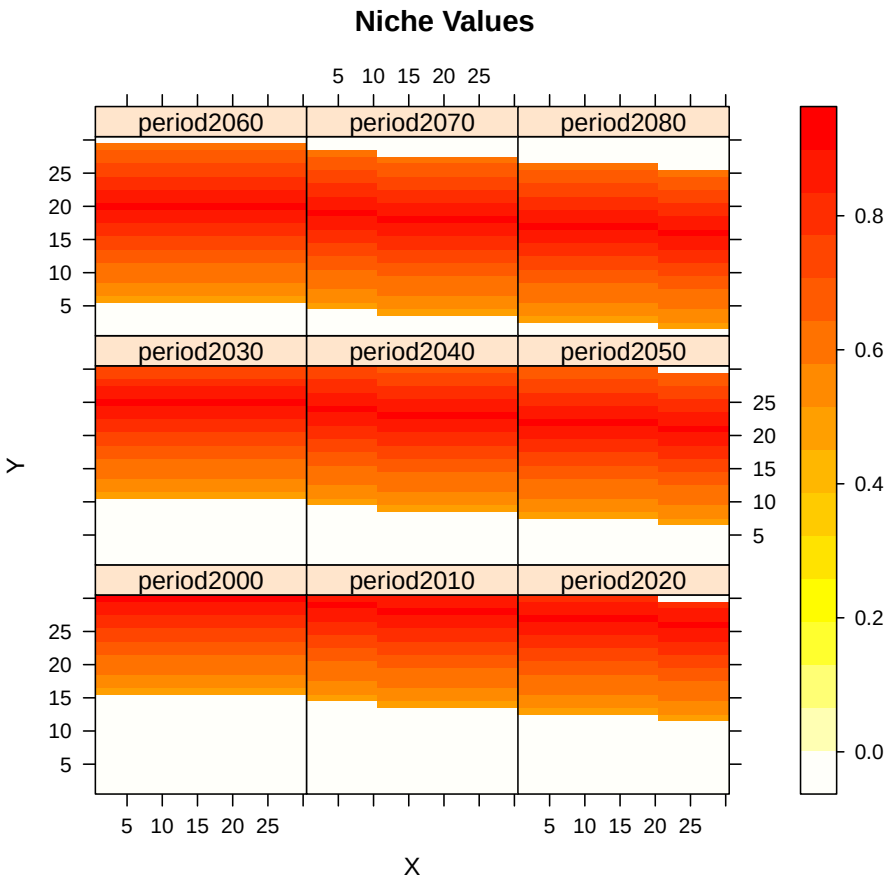


Figure 4: Niche values at each modelled time period.

```
> library(popbio)
> data(hudvrs)
> data(hudsonia)
> matrices_mine <- cbind(meanmatrix = as.vector(hudmxdef(hudvrs$mean)),
+     sapply(hudsonia, unlist))
> colnames(matrices_mine) <- c("Reference_matrix",
+     "Mx1", "Mx2", "Mx3", "Mx4")
> stages_mine <- colnames(hudsonia$A85)
```

Make sure that the `matrix` command produces the correct matrix (dependent on the original order of the matrix elements, by row or by column).

```
> matrix(matrices_mine[, "Reference_matrix"], ncol = length(stages_mine),
+     byrow = FALSE)
```

```
       [,1]   [,2]      [,3]       [,4]        [,5]
[1,] 0.4995 0.0000 4.5782000 12.1425000 22.31670000
[2,] 0.0004 0.0000 0.0033000  0.0088000  0.01620000
[3,] 0.0000 0.4773 0.5988709  0.2025511  0.07961523
[4,] 0.0000 0.0000 0.1891171  0.4726192  0.10947094
[5,] 0.0000 0.0000 0.0000000  0.2662100  0.55730661
[6,] 0.0000 0.0000 0.0000000  0.0231487  0.24879760
            [,6]
[1,] 50.18950000
[2,]  0.03640000
[3,]  0.00000000
[4,]  0.06865331
[5,]  0.17653707
[6,]  0.73557114
```

Which stages should be counted when calculating population sizes. In this case, we do not count the seed stage when calculating population sizes. It can also be "all_ stages" for all stages.

```
> sumweight_mine <- c(0, 1, 1, 1, 1, 1)
```

```
[1] 0 1 1 1 1 1
```

```
> proportion_initial_mine <- c(0.9818098089, 0.0006907668,
+     0.0069076675, 0.0036840893, 0.0057563896,
+     0.0011512779)
```

```
[1] 0.9818098089 0.0006907668 0.0069076675 0.0036840893
[5] 0.0057563896 0.0011512779
```

Density of individuals (all stages, including seeds) in patches. Same units as area sizes as in Populations_ mine (sometimes lat/long)

```
> density_individuals_mine <- 20000
```

```
[1] 20000
```

When populations are over this (a multiple) above compared to the population with the largest original population they will reach K carrying capacity, and will be reduced to K. If NULL, there is no density dependence effects in the model.

```
> K_mine <- 100
```

```
[1] 100
```

Variances in vital rates

```
> matrices_var_mine <- matrix(0.01, ncol = 1, nrow = nrow(matrices_mine),
+     dimnames = list(NULL, "sd"))
```

Possible arguments: either FALSE (no), "all" (all nonzero transition probabilities) or a vector of affected stages.

```
> transition_affected_niche_mine <- c(1, 3)
```

```
[1] 1 3
```

Either TRUE(all nonzero stages) or a vector of affected stages (as above).

```
> transition_affected_env_mine <- "all"
```

```
[1] "all"
```

Either TRUE("all_nonzero_stages") or a vector of affected stages.

```
> transition_affected_demogr_mine <- "all"
```

```
[1] "all"
```

Do you want the environmental stochasticity to be normal or lognormal? Indicate "normal" or "lognormal" here. The default is a "normal" distribution.

```
> env_stochas_type_mine <- "normal"
```

```
[1] "normal"
```

With which probability should the scenario matrices should be drawn? Should be a vector of two numbers. Default is equal probability.

```
> prob_scenario_mine <- c(0.5, 0.5)
```

```
[1] 0.5 0.5
```

Temporal autocorrelation, noise. Is the change in probability that the same matrix as the last iteration will be chosen the next time period. A noise value of 1 is completely random. $0 <$ noise < 2.

```
> noise_mine <- 0.95
```

```
[1] 0.95
```

3.3 Specify dispersal

Fraction of seeds that disperse short distance, beyond patch borders. Dispersal takes palce to the 8 contigous cells. If this is zero, no SDD is modelled.

```
> fraction_SDD_mine <- 0.05
```

```
[1] 0.05
```

Fraction of seeds that disperse long distance, beyond patch borders. The direction of seeds is deterined by the distances between populations. If this is zero, no LDD is modelled.

```
> fraction_LDD_mine <- 0.05
```

```
[1] 0.05
```

Constants for LDD dispersal kernel in the format: c(a, b, c, Distmax) If the distance between populations is larger than Distmax (in kilometers, there will be no dispersal between patches.

```
> dispersal_constants_mine <- c(0.7, 0.7, 0.1, 200)
```

```
[1]   0.7   0.7   0.1 200.0
```

3.4 Projection information

Length in years of each time period

```
> no_yrs_mine <- 10
```

```
[1] 10
```

3.5 Run setup function

Now we have all the information we need to run the setup function, and create our species object. Below is the complete function, with all the objects specified. The setup function generates one species object with all the necessary information for the modeling in the workspace. It also checks the consistency of data, and prints error messages. The species object is also saved in the working directory, as an R-object (`.rda`).

```
> demoniche_setup(modelname = "Hmontana", Populations = Populations_mine,
+     Nichemap = Nichemap_mine, matrices = matrices_mine,
+     matrices_var = matrices_var_mine, noise = noise_mine,
+     prob_scenario = prob_scenario_mine, stages = stages_mine,
+     proportion_initial = proportion_initial_mine,
+     density_individuals = density_individuals_mine,
+     fraction_LDD = 0.05, fraction_SDD = fraction_SDD_mine,
+     dispersal_constants = dispersal_constants_mine,
+     transition_affected_niche = transition_affected_niche_mine,
+     transition_affected_demogr = transition_affected_demogr_mine,
+     transition_affected_env = transition_affected_env_mine,
+     env_stochas_type = env_stochas_type_mine,
+     no_yrs = no_yrs_mine, K = K_mine, sumweight = sumweight_mine)
```

We have now re-created the species object which is called the 'modelname' argument that we specified. As above, this can be used in the `demoniche_model` function (section 2.3).

Note that we defined the fraction of seeds dispersing long distances directly in the function, instead of `fraction_LDD_mine <- 0.05` and then defining the object in the setup function: `demoniche_setup(..., fraction_LDD = fraction_LDD_mine)`, we simply wrote the value directly `demoniche_setup(..., fraction_LDD = 0.05)`.

4 Other

Related models: Vortex (Lacy et al. 1993), ULM (Legendre and Clobert 1995) RamasGIS/Metapop (Akaçkaya and Root 2005), Prunus (Sebert-Cuvillier et al. 2009), BioMove (Midgely et al. 2010)

Related R-packages: `popbio`, `Rramas`

Recommended packages: `sp` to load shapefiles, `BIOMOD` and `dismo` to carry out Species Distribution Modelling

5 Acknowledgements

Thanks to Rebecca Swab and Regan Early for comments and advice about demographic modelling. Many thanks to François Guilhaumon for generously sharing knowledge about coding and package-building in R. Thanks to popbio and BIOMOD packages for inspiration. Early versions of this model were made at the Center for Spatial Analysis, Universidad Mayor de San Andres, Bolivia.

6 Citations

Akçakaya H.R., Burgman M.A., Kindvall O., Wood C.C, Sjogren-Gulve P., Hatfield J.S. and McCarthy M.A. (Eds) 2002. Species conservation and Management: Case Studies. Oxford University Press.

Akçakaya, H. R. and Root, W. T. 2005. RAMAS GIS : linking spatial data with population viability analysis, version 5.0. Setauket, NY: Applied Biomathematics.

Caswell, H. 1989. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates, Sunderland, Massachusetts.

de la Cruz, M. 2010. Rramas: Matrix population models. R package version 0.1-0. http://CRAN.R-project.org/package=Rramas

Gross, K., Iii, J.R.L., Frost, C.C., Morris, W.F., 1998. Modeling Controlled Burning and Trampling Reduction for Conservation of Hudsonia montana. Conservation Biology 12, 1291-1301.

Keith, D.A., Akçakaya, H.R., Thuiller, W., Midgley, G.F., Pearson, R.G., Phillips, S.J., Regan, H.M., Araujo, M.B., Rebelo, T.G., 2008. Predicting extinction risks under climate change: coupling stochastic population models with dynamic bioclimatic habitat models. Biology Letters 4, 560-563.

Lacy, R. C. 1993. VORTEX: a computer simulation model for population viability analysis. Wildlife Research 20: 45-65.

Legendre S. and Clobert J. 1995. ULM, a software for conservation and evolutionary biologists. Journal of Applied Statistics 22: 817-834.

Midgley, G.F., Davies, I.D., Albert, C.H., Altwegg, R., Hannah, L., Hughes, G.O., OHalloran, L.R., Seo, C., Thorne, J.H., Thuiller, W., 2010. BioMove - an integrated platform simulating the dynamic response of species to environmental change. Ecography

Morris, W.F., and D.F. Doak. 2002. Quantitative Conservation Biology. Theory and Practice of Population Viability Analysis. Sinauer Associates, Sunderland, Massachusetts.

Pagel, J., Schurr, F.M., 2011. Forecasting species ranges by statistical estimation of ecological niches and spatial population dynamics. Global Ecology and Biogeography.

R Development Core Team (2010). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Sebert-Cuvillier, E., Simonet, M., Simon-Goyheneche, V., Paccaut, F., Goubet, O., Decocq, G., 2009. PRUNUS: a spatially explicit demographic model to study plant invasions in stochastic, heterogeneous environments. Biol Invasions 12, 1183-1206.

Sjogren-Gulve, P. and T. Ebenhard. Eds. 2000. The Use of Population Viability Analyses in Conservation Planning. Ecological Bulletin 48: 9-21.

Stubben, C.J. and Milligan, B.G. 2007. Estimating and Analyzing Demographic Models Using the popbio Package in R. Journal of Statistical Software 22:11.

Thuiller, W., Lafourcade, B., Engler, R., Araujo, M., 2009. BIOMOD - a platform for ensemble forecasting of species distributions. ECOGRAPHY 32, 369-373.