

demoniche

Hedvig Nenzén

September 7, 2011

Contents

1	Introduction	2
2	How to use demoniche	2
2.1	Install the package	3
2.2	Loading data supplied with package	3
2.3	Modelling	4
2.4	Analyse data	5
3	Setting up our own data	11
3.1	Geographic information	12
3.1.1	Original population distribution	12
3.1.2	Niche data	12
3.2	Demographic information	13
3.2.1	Transition matrices	13
3.2.2	Stages	14
3.2.3	Weight of stages	14
3.2.4	Stages affected by Niche values	15
3.2.5	Stages affected by environmental stochasticity	15
3.2.6	Stages affected by demographic stochasticity	15
3.2.7	Type of environmental stochasticity	15
3.2.8	Coefficients of variation in vital rates	15
3.2.9	Initial proportion	16
3.2.10	Population density	16
3.2.11	Density dependence	16
3.2.12	Scenario probabilities	16
3.2.13	Noise	16
3.2.14	Short distance dispersal	17
3.2.15	Long distance dispersal	17
3.2.16	Dispersal kernel	17
3.3	Run <code>setup</code> function	17
4	Modifying demoniche functions	18
5	Other resources	19
6	Acknowledgements	19
7	References	19

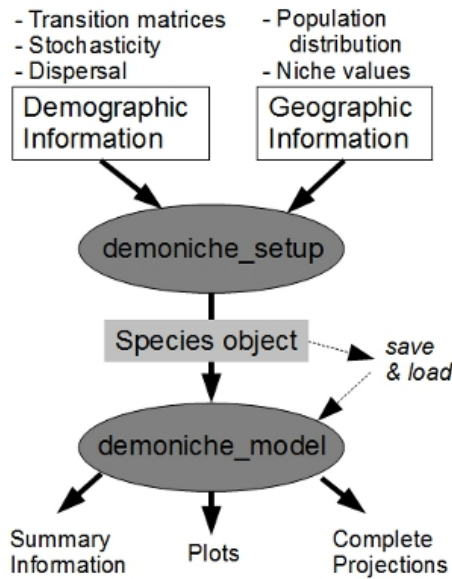


Figure 1: Simplified schema of the *demoniche* framework. First the user enters demographic and geographic information into the setup function. The model function runs the demographic modelling on the species object.

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 influenced by scenarios of environmental and human impacts.

The *demoniche* package is offered without any guarantees, and we encourage users to examine the code to learn what is modelled, and to adapt it to their own needs.

The main steps to running a model are as follows:

- Load or write the information (demographical, geographical) of the species that is being modelled
- Create a species object with `demoniche_setup` function, which contains all the information about the species
- Run the `demoniche_model` function on the species object
- Analyse the results

2 How to use demoniche

First install R. We also recommend that users are familiar with the R environment. This can be gained from, for example www.cran.r-project.org, where the program can be downloaded too.

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 package and manual are often updated, so please download an up-dated package often.

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. This makes the two main functions *demoniche_model* and *demoniche_setup* available (Figure 1). The *demoniche_model* function runs two internal functions, *demoniche_population* that carries out demographic modelling in each population and year, and *demoniche_dispersal* that calculates the dispersal if selected.

```
> library(demoniche)
```

The code used 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* in Windows, 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 *Hmontana* 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 with 26 items. We can examine separate items 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 ...
```


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. If you only have one transition matrix, and run models without stochasticity, the mean and EMA will be the same, and the standard deviation will be zero, regardless of numbers of repetitions. The third dimension of the array are the results for 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" "year7"
 [8] "year8" "year9" "year10" "year11" "year12" "year13" "year14"
[15] "year15" "year16" "year17" "year18" "year19" "year20" "year21"
[22] "year22" "year23" "year24" "year25" "year26" "year27" "year28"
[29] "year29" "year30" "year31" "year32" "year33" "year34" "year35"
[36] "year36" "year37" "year38" "year39" "year40" "year41" "year42"
[43] "year43" "year44" "year45" "year46" "year47" "year48" "year49"
[50] "year50" "year51" "year52" "year53" "year54" "year55" "year56"
[57] "year57" "year58" "year59" "year60" "year61" "year62" "year63"
[64] "year64" "year65" "year66" "year67" "year68" "year69" "year70"
[71] "year71" "year72" "year73" "year74" "year75" "year76" "year77"
[78] "year78" "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 Minimum Abundance from simulations with the treatment matrix Mx4:

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

      year1      year2      year3      year4      year5      year6      year7
75978    84339    99256   145146   167562   222134   249965
      year8      year9     year10     year11     year12     year13     year14
376847   415388   484113   539611   683136   787876  1065679
```

year15	year16	year17	year18	year19	year20	year21
1354349	1598061	1872626	2138504	2394529	3891905	4601321
year22	year23	year24	year25	year26	year27	year28
5401571	8321111	10126403	10938141	13043852	16073021	21268366
year29	year30	year31	year32	year33	year34	year35
23611784	32029534	34458835	42769071	47767202	52110359	52822241
year36	year37	year38	year39	year40	year41	year42
57970252	70210318	73281611	87537811	86807596	90275647	85246431
year43	year44	year45	year46	year47	year48	year49
95854930	98342804	103754031	99987706	101324695	109241251	108208091
year50	year51	year52	year53	year54	year55	year56
111684567	111540271	105514532	111060117	109351296	110225632	108767840
year57	year58	year59	year60	year61	year62	year63
103419228	101732143	98294624	106480405	108491408	111274909	106422856
year64	year65	year66	year67	year68	year69	year70
102616051	105881060	109752657	103322776	109387058	104335352	109396434
year71	year72	year73	year74	year75	year76	year77
105189145	109572284	107280293	110712774	107935354	106362856	105696405
year78	year79	year80	year81	year82	year83	year84
110192294	109304556	104929251	102097867	111186937	113663225	110302066
year85	year86	year87	year88	year89	year90	
107871572	109291872	104278906	112751926	112594141	104534849	

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"))
```

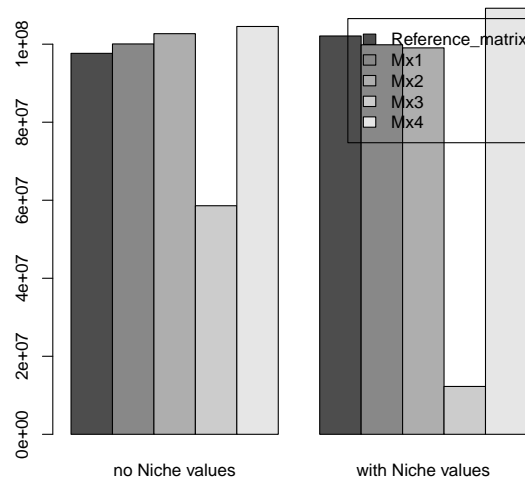


Figure 2: Population size (number of individuals) for simulations under each treatment matrix scenario of human land use, in the last year of simulation (year 90). With climate change there was a lower predicted population size

Apart from the summary object, the results from each simulation are saved in the folder with the specified name, in the working directory. If any of the files are open while we run new simulations, they will not be updated (i.e. automatically overwrites files, but cannot do it if files are open). We can look at what results 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 growth rate (λ , the first eigenvalue) of the treatment matrix (not the reference matrix, function from `popbio` package), the stochastic λ from the two matrices (50.000 time intervals, function from `popbio` package). *Some of these results 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 ...
```


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, 3.1

- 3.1.1 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).
- 3.1.2 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). If no Niche values are made, a background grid is automatically created.

Demographic information, 3.2

- 3.2.1 Age(Leslie)- or stage(Lefkovitch)-based transition matrix. Various treatment or catastrophe 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-defined probability, 3.2.12.
- 3.2.2 Name of each life stage
- 3.2.3 Weight of stages
- 3.2.4, 3.2.5 and 3.2.6 Which stages in the transition matrix which are affected by Niche values, and environmental and demographic stochasticity.
- 3.2.7 Type of distribution of the environmental stochasticity
- 3.2.8 Coefficients of variation in vital rates. Optional coefficient of variation matrix for each value in the transition matrix.
- 3.2.9 The initial proportion of individuals in each stage.
- 3.2.10 Initial population density
- 3.2.11 Density dependence. 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.
- 3.2.13 Noise
- 3.2.14 and 3.2.15 The fraction that seeds that undergo short and long distance dispersal.
- 3.2.16 Dispersal kernel

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 simply write the value `demoniche_setup(... no_years = 10)`. This is easier when values are short.

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

3.1 Geographic information

3.1.1 Original population distribution

This should be one file with ID of patches, 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

3.1.2 Niche data

This can be a regular grid comprising the locations of the populations. In this case we are using predicted probabilities of presence under climate change (from a GLM for example). If not supplied, a background grid is created.

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

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

With the `lattice` package we can plot the Niche values and see how the most suitable areas move towards the south, Figure 3.1.2.

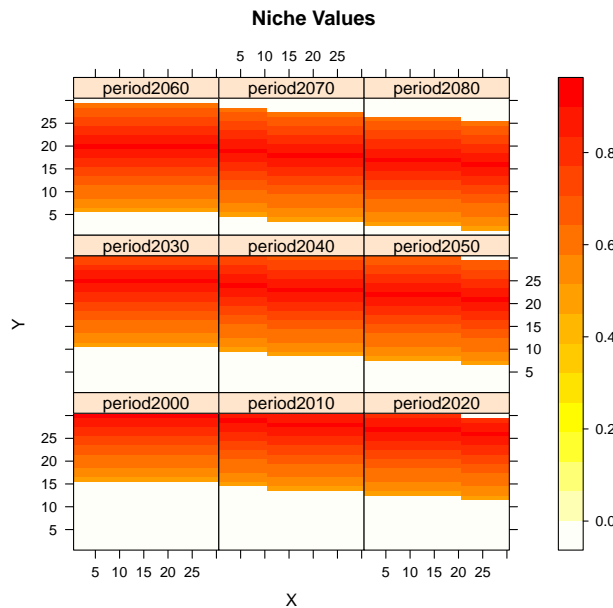


Figure 4: Niche values at each modelled time period.

```
> niche_formulas <- as.formula(paste(paste(colnames(Nichemap_mine)[-c(1:3)],
+   collapse = "+"), "X+Y", sep = "~"))

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"))

Length in years of each time period.

> no_yrs_mine <- 10

[1] 10
```

3.2 Demographic information

3.2.1 Transition matrices

A matrix/data frame of transition 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, in which case the EMA and mean population will be the same, and the standard deviation NA (deterministic modelling)

Here we use data from the `popbio` package. The transition probabilities (matrix entries, a_{ij}) were originally in a square matrix format, so we unlist each matrix so that each matrix is now one column in the matrix object.

```
> library(popbio)
> data(hudvrs)
> data(hudsonia)
> matrices_mine <- cbind(meanmatrix = as.vector(hudmxdef(hudvrs$mean)),
+   sapply(hudsonia, unlist))
> head(matrices_mine)
```

```

      meanmatrix   A85   A86   A87   A88
[1,]      0.4995 0.4995 0.4995 0.4995 0.4995
[2,]      0.0004 0.0004 0.0004 0.0003 0.0003
[3,]      0.0000 0.0000 0.0000 0.0000 0.0000
[4,]      0.0000 0.0000 0.0000 0.0000 0.0000
[5,]      0.0000 0.0000 0.0000 0.0000 0.0000
[6,]      0.0000 0.0000 0.0000 0.0000 0.0000

```

```

> colnames(matrices_mine) <- c("Reference_matrix", "Mx1",
+   "Mx2", "Mx3", "Mx4")

```

We see that the `matrices` object is a matrix with each original transition matrix as a column and the names of the matrices as the column names (that do not agree with the original matrix names in this case).

The order of the transition probabilities is important. Make sure that running the function `matrix()` function with the option `byrow = FALSE` on one of the matrix columns reproduces the correct matrix. This depends on the original order of the matrix elements, if it was by row or by column. Below we see that the seed (fertility) stage is on the top row, which is correct. When later we set which transition probabilities a certain type of stochasticity should effect, it is these row numbers that we mean. For example, if we set effects on matrix entry 1, is refers to top left matrix element ([1,1]) of the matrix.

```

> matrix(matrices_mine[, "Reference_matrix"], ncol = 6, byrow = FALSE)

```

```

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

```

3.2.2 Stages

In this case, the column names of the original matrix are used to make the `stage` object which has the names of each life stage.

```

> stages_mine <- colnames(hudsonia$A85)

[1] "seed"      "seedlings" "tiny"      "small"     "medium"
[6] "large"

```

3.2.3 Weight of stages

We can specify which life stages should be counted when calculating population sizes, and to which stages that the density dependent growth should be applied to. In this example, we do not count the seed stage when calculating population sizes, and the seeds are not subject to density dependent growth. It can also be "all_stages" if all stages should be included.

```

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

[1] 0 1 1 1 1 1

```

3.2.4 Stages affected by Niche values

Here we specify which transition probabilities (matrix elements, a_{ij}) should be affected by the Niche values. We can specify the character string "all" which means that all non-zero stages are affected. Alternatively, we enter the matrix stages as a vector of numbers which corresponds to the correct matrix element. In this case, we set Niche values to affect rows 1 and 3 of the **matrices** object. We know that the first and third row in our **matrices** object refer to the probability of a seed surviving in the seed-bank and the probability of a seed becoming a seedling.

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

[1] 1 3

> matrices_mine[transition_affected_niche_mine, 1]

[1] 0.4995 0.0000
```

3.2.5 Stages affected by environmental stochasticity

Here we specify which transition probabilities (matrix elements, a_{ij}) should be affected by and environmental and demographic stochasticity. To use no stochasticity and carry out deterministic modelling, we set this to **FALSE** (default). This vector refers to the affected transition stages in the same way as the Niche values, 3.2.4.

Possible arguments: "all" (all nonzero transition probabilities) or a vector of affected stages.

```
> transition_affected_env_mine <- "all"

[1] "all"
```

3.2.6 Stages affected by demographic stochasticity

Possible arguments: either "all" (all nonzero transition probabilities) or a vector of affected stages.

```
> transition_affected_demogr_mine <- "all"

[1] "all"
```

3.2.7 Type of environmental stochasticity

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"
```

3.2.8 Coefficients of variation in vital rates

Coefficients of variation in vital rates which are used to set how variable the environmental and demographic stochasticity should be. In this case they are set as 0.01 for all matrix elements.

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

3.2.9 Initial proportion

This is the initial proportion of individuals in all stages, and should be a vector of the same length as the number of stages, and should add to 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 0.0057563896  
[6] 0.0011512779
```

3.2.10 Population density

Density of individuals (all stages, including seeds) in each grid cell. This number is automatically multiplied by the column named 'area Populations_mine, and should be in the same units as the area in the Populations object. This can be a single value, which is then used for all populations, or a vector if densities in each population are known. Multiply by niche values to get like ramans. This density value only affects the initial abundances, and not future projections.

```
> density_individuals_mine <- 20000  
[1] 20000
```

3.2.11 Density dependence

Density dependent growth. When populations are over this (a multiple) above compared to the population with the largest original population they will reach carrying capacity, and will be reduced to the value of K. If it is NULL, there is no carrying capacity limit in the simulations. The demoniche model also provides the alternative of defining a carrying capacity for each population, in this case the K should be a vector of values. These values could be defined by various means, for example from multiplying a density by the Nichevalues, but this has to be done outside the model.

```
> K_mine <- 100  
[1] 100
```

3.2.12 Scenario probabilities

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

```
> prob_scenario_mine <- c(0.5, 0.5)  
[1] 0.5 0.5
```

3.2.13 Noise

An option of specifying temporal autocorrelation (noise) in the matrix selections also exists, so that the probability of choosing the same matrix in subsequent years is decreased or increased. 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 (default). $0 < \text{noise} < 2$.

```
> noise_mine <- 0.95  
[1] 0.95
```


3.2.14 Short distance dispersal

Dispersal between populations can be either short (with dispersal to the eight contiguous cells) or long-distance dispersal. Fraction of seeds that disperse short distance, beyond patch borders. Dispersal takes place to the 8 contiguous cells. The seeds that disperse to these cells will be subjected to the same matrix multiplication as all cells. If this is zero, no SDD is modelled.

```
> fraction_SDD_mine <- 0.05
```

```
[1] 0.05
```

3.2.15 Long distance dispersal

The long-distance dispersal is modelled by an exponential dispersal kernel with user-defined constants according to the formula $P = a * \exp(D^{(c/b)})$, with D = distance in kilometers between populations. These constants describe the shape of the species dispersal probability curve, and define a maximum distance beyond which dispersal is impossible. Long- and short-distance dispersal can be applied separately or together; alternatively, simulations without dispersal can be run.

The direction of seeds is determined by the distances between populations. If this is zero, no LDD is modelled.

```
> fraction_LDD_mine <- 0.05
```

```
[1] 0.05
```

3.2.16 Dispersal kernel

Constants for LDD dispersal kernel as a vector 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
```

3.3 Run setup function

Now we have defined all the information we need to run the setup function, and we can examine what objects we have in the work space. First we remove objects created earlier.

```
> rm(CC_nodispersal, eigen_results, Hmontana, hudsonia, hudvrs,  
+     niche_formulas, noCC_nodispersal)
```

```
NULL
```

```
> ls()
```

```
[1] "density_individuals_mine"      "dispersal_constants_mine"  
[3] "env_stochas_type_mine"        "fraction_LDD_mine"  
[5] "fraction_SDD_mine"            "K_mine"  
[7] "matrices_mine"                "matrices_var_mine"  
[9] "Nichemap_mine"                "noise_mine"  
[11] "no_yrs_mine"                  "Populations_mine"  
[13] "prob_scenario_mine"           "proportion_initial_mine"  
[15] "stages_mine"                  "sumweight_mine"  
[17] "transition_affected_demogr_mine" "transition_affected_env_mine"  
[19] "transition_affected_niche_mine"
```

We specify these objects to each argument in the `demoniche_setup()` function and create the species object. Below is the complete function, with all the arguments 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 any error messages. The order of the arguments does not matter, as long as all are present. The species object is also saved in the working directory, as an R-object (`.rda`) which can be loaded at a later time, or shared.

```
> 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)`. Either way works.

4 Modifying demoniche functions

We encourage users to inspect the code in the `demoniche` functions, to find out how the algorithm is set up. By typing in only the function names in R (i.e. `demoniche_setup` directly into the console, without parentheses) the algorithm is printed.

If the user wishes to change how a function is carried out, this is possible and encouraged. Briefly, one copies the code to a new R-script, makes the desired changes to the function definition, and then reads the new function into the workspace. Then the modified function can be used as the original package function.

For example, say we wish to change the short-distance dispersal function. Currently propagules can spread to the 8 neighboring grid cells, as the function now is defined (the neighborhood index in the species object). In this case, whenever two patches are 1.0 or 1.4 degrees apart, we know that they are contiguous or diagonal from each other and propagules flow between them.

```
> Hmontana$neigh_index
```

```
[1] 1.0 1.4
```

The neighborhood index is defined in the `demoniche_setup` function (around line 100). First a distance matrix between all locations in the Niche values file is calculated, then the second and third largest values are saved in the species object. The behaviour of this code can be examined with the original coordinates from the Niche values.

```
dist_latlong <- round(as.matrix(dist(Niche_ID[,2:3])), 1)
neigh_index <- sort(unique(as.numeric(dist_latlong)))[2:3]
```

We wish to modify this, so that migrating propagules only can spread to the 4 neighboring cells, directly contiguous cells. We would have to change the function so that only the first value is saved in the neighborhood index, i.e. only 1 in this case, in the `demoniche_setup` function.

To make our own function, we first type: `demoniche_setup` into the console. The current function definition is returned to us, starting with `> demoniche_setup function(modelname, Populations,, etc.` We copy everything to a new R script, taking care to copy the last `}` which ends the function. We change the top of the function so that it becomes a standard function definition: `demoniche_setup_fourneighbors <- function(modelname, Populations, etc.` Here we can also change the actual name of the function, to distinguish it from the original function: maybe `demoniche_setup_fourneighbors`. We then change the lines in the function that we wish to modify: `neigh_index <- sort(unique(as.numeric(dist_latlong)))[2]` to only save the second number. If we run this entire function we have defined a new function called `demoniche_setup_fourneighbors` (check it!).

It is sometimes easier to save an R-script with only the function, like for example `fcf_fourneighbors.R`. Then the command `source("fcf_fourneighbors.R")` can be used which reads the function into the working space (just like when loading a package).

To run our modified function, we do the same thing as when running the original setup function but with the new function name: `demoniche_setup_fourneighbors(modelname = "Hmontana", Populations = Populations_mine,, etc.` This will create a species object in which the `Hmontana$neigh_index` only has one number. When we run the model function on this object, dispersal will only take modelled to the four contiguous cells.

5 Other resources

Related models: Vortex (Lacy et al. 1993), ULM (Legendre and Clobert 1995) RamasGIS/Metapop (Akçakaya and Root 2005), Patch (Schumaker 1998, Schumaker et al. 2004), TreeMig (Lischke et al. 2006), Prunus (Sebert-Cuvillier et al. 2009), BioMove (Midgely et al. 2010).

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

Recommended packages: `sp` to load shapefiles, `BIOMOD` and `dismo` to carry out Species Distribution Modelling, `SDMTools` to calculate fragmentation indices.

6 Acknowledgements

Many thanks to Rebecca Swab for patiently and greatly improving the user-friendliness of the model. Many thanks to François Guilhaumon for generously sharing knowledge about coding and package-building in R. Thanks to Regan Early for comments and advice about demographic modelling. 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 Andrés, Bolivia.

7 References

- 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.

- Hyndman, R. J. demography: Forecasting mortality, fertility, migration and population data at: <http://robjhyndman.com/software/demography>
- Keith, D.A., Akçakaya, H.R., Thuiller, W., Midgley, G.F., Pearson, R.G., Phillips, S.J., Regan, H.M., Araújo, 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.
- Lischke, H., Zimmermann, N.E., Bolliger, J., Rickebusch, S., Löffler, T.J. 2006 TreeMig: a forest-landscape model for simulating spatio-temporal patterns from stand to landscape scale. *Ecological Modelling* 199: 409-420.
- 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* 33: 612-616.
- 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*. DOI: 10.1111/j.1466-8238.2011.00663.x
- 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>.
- Schumaker, N.H. 1998. *A Users Guide to the PATCH Model*. EPA/600/R-98/135, U.S. Environmental Protection Agency, Environmental Research Laboratory, Corvallis, Oregon, 120.
- Schumaker, N.H., Ernst, D., White, J., Baker, P., Haggerty, 2004. Projecting wildlife responses to Alternative Future Landscapes in Oregon's Willamette Basin. *Ecological Applications*, 14, 381-400.
- 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. *Biological 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., Araújo, M., 2009. BIOMOD - a platform for ensemble forecasting of species distributions. *Ecography* 32: 369-373.