MigClim user guide (for R)

Robin Engler, April 2012.

Table of Contents

About this user guide: this is version 1.1.0 of the MigClim R user guide.

The MigClim R package and this user guide are distributed in the hope that they will be useful, but come without any warranty, expressed or implied. MigClim, this user guide as well as the accompanying test data all come "as is" and without any warranty of support from its authors. This being said, I'm happy to hear about your experience, bug reports, suggestions, etc... and will try to help you within the limits of my spare time. You can contact me by e-mail at: Robin.Engler [-at-] gmail.com.

Part 1: Introduction and model overview.

Why use MigClim ?

When using species distribution models (also known as habitat suitability models or resource selection functions) to identify changes in a species' potential distribution, one recurrent problem is to account for species dispersal limitations. Here are two typical examples illustrating this issue.

The first example deals with projecting the potential distribution of a species into the future under a climate change scenario. When simply using the projections obtained from habitat suitability models to predict changes in distribution, we implicitly assume unlimited dispersal (i.e. the species can occupy any habitat that is suitable, regardless of its location). There are however many elements that can challenge this assumption, and impede a species to reach all of its potential habitat. For instance, the increasing fragmentation of the landscape could be a barrier to species dispersal, or at least slow down its dispersal rate. Maybe some of the suitable habitats are located too far away from the current populations of the species, and are thus out of reach. Maybe dispersing to some of the suitable habitat is hindered by the presence of a physical barrier (e.g. a river or a highway might impede the dispersal of some animal species across it).

Similarly if we attempt to model the potential distribution of an invasive species, dispersal limitations could also be an important parameter to take into account. In the case of an invasive species, the most important question is maybe not whether all potentially suitable locations will eventually become colonized or not (generally invasive species are rather good at dispersing), but when they might become colonized and which is the most likely route that the species will take to spread through the landscape. Again, accounting for dispersal appears as an important issue in this context.

Both of these examples illustrate the need to account for dispersal limitation when projecting changes in species distribution. The MigClim model has been developed to address this kind of issues. It can be thought off as a supplementary component of a species distribution model, that acts between the habitat suitability predictions and the final potential distribution of a species, by restricting this potential distribution through dispersal limitations (Fig. 1).

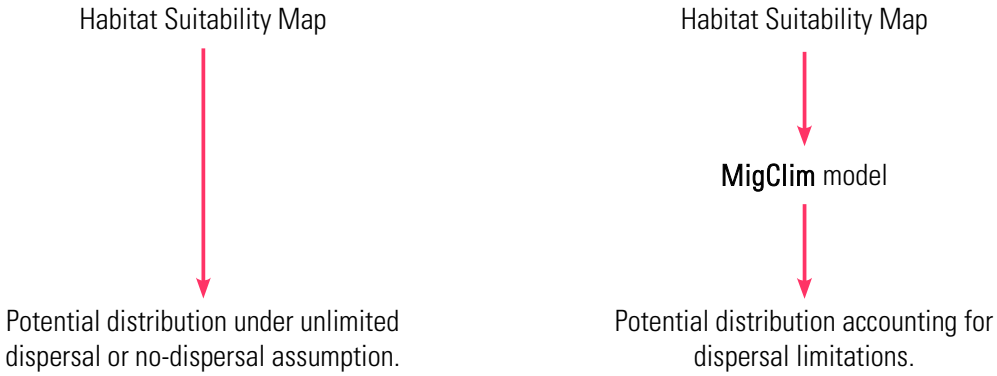


Figure 1. MigClim allows restricting changes in a species' potential distribution through dispersal limitations.

The MigClim R package is a function library for the open source R software (www.r-project.org) that enables the implementation of species-specific dispersal constraints into projections of species distribution models under environmental change and/or landscape fragmentation scenarios. The model is based on a cellular automaton and the basic modeling unit is a cell that is inhabited or not. Model parameters include dispersal distance and kernel, long distance dispersal, barriers to dispersal, propagule production potential and habitat invasibility. The MigClim R package has been designed to be highly flexible in the parameter values that it accepts, and to offer good compatibility with existing species

distribution modeling software (e.g. BIOMOD or MAXENT). Possible applications include the projection of species distribution under environmental change conditions or modeling the spread of invasive species.

Note that, since the idea behind MIGCLIM is to provide users with a relatively easy to calibrate and flexible model that can adapt to many species, MIGCLIM does currently not implement any advanced population dynamics parameters such as number of individuals per cell or number of seeds produced per individual.

To get a better idea of why the MIGCLIM model was developed, how it works and how it can be applied to a large number of species, I recommend reading the following (highly entertaining ;-) scientific publications:

→ Engler R., Hordijk W. and Guisan A. The MIGCLIM R package – seamless integration of dispersal constraints into projections of species distribution models. *Ecography,* in review.

→ Engler R. and Guisan A., 2009. MIGCLIM: Predicting plant distribution and dispersal in a changing climate. *Diversity and Distributions*, **15** (4), 590-601.

→ Engler R., Randin C.F., Vittoz P., Czáka T., Beniston M., Zimmermann N.E. and Guisan A, 2009. Predicting future distributions of mountain plants under climate change: does dispersal capacity matter? *Ecography*, **32** (1), 34-45.

Engler et al. (in review) explains the purpose of the MIGCLIM R package and how to use it (although not in as much details as the present user guide). Engler and Guisan (2009) explains how the MIGCLIM model works and gives an example of application for two semi-virtual species. Engler *et al.* (2009) provides an example of the application of MIGCLIM to a large number of species and could thus give you some indications on how to calibrate the model for a large number of species with limited available data.

Note: Engler and Guisan (2009) actually presents the old version of MIGCLIM that was implemented within the ArcGIS software rather than the R software. While the core of the model has not changed much, there has been a few changes in options and input formats.

Although I'm no longer maintaining the ArcGIS version of MIGCLIM, it is still available on request (e-mail me – see my address on the title page of this document) and should work well with versions 9.1, 9.2 and 9.3 of ArcGIS.

If you use MIGCLIM, please cite it as follows:

→ Engler R., Hordijk W. and Guisan A. The MIGCLIM R package – seamless integration of dispersal constraints into projections of species distribution models. *Ecography,* in review.

Frequently asked (and important) questions

Does MigClim generate its own habitat suitability data? If not where can I get it from?

No, MigClim does not generate habitat suitability maps itself, and you have thus to generate your habitat suitability data using another software before being able to use MigClim.

The good news though is that there are already very efficient tools out there to generate such habitat suitability data, for instance the Biomod package that also runs in R (see Thuiller, W., Lafourcade, B., Engler, R. and Araújo, M.B. Biomod - A platform for ensemble forecasting of species distributions. Ecography, 32, 369-373, http://r-forge.r-project.org/projects/biomod).

The even better news is that MigClim was designed to integrate seamlessly with these existing modeling tools (in particular Biomod), so that you can use their outputs as MigClim inputs with minimal (or no) effort.

> *Important: MigClim habitat suitability data must be given in the range [0:1000] and must be integer numbers (floating point numbers are not accepted).*

I'm a Biomod addict, can I use Biomod outputs with MigClim?

Yes, you certainly can. In fact the MigClim inputs have been designed to have the same scale of habitat suitability values as Biomod (integer numbers from 0 to 1000), so you can use your Biomod outputs right away. If you wish to work with binary values (habitat is suitable or not), you don't have to worry about reclassifying your data: simply indicate your reclassification threshold and MigClim will reclassify your continuous projections.

If you wish to work without the thresholding option, then the values of habitat suitability (in the range 0:1000) are used as a cell "invasibility index" and are interpreted as an absolute probability of presence conditional on the species dispersing to the cell (see also equation 1 in this document). In other words, all other things being equal, a cell with habitat suitability of 600 is twice as likely to be colonized than a cell with habitat suitability of 300. Note that depending on how you calibrated your model, the predictions are not necessarily absolute probabilities. If they are not, then they you will have either to rescale them, or to use the thresholding option.

In terms of habitat suitability values input format, MigClim accepts both data frame and raster formats, so whatever format you chose to carry-out your projections in Biomod, your output will be compatible with MigClim.

I'm Maxent-dependent, is that compatible with MigClim?

Yes it is, although it will require a few extra steps. In its 'logistic' format, Maxent outputs its results in a scale form [0:1], rather than [0:1000]. So you should multiply the Maxent's 'logistic' output values by 1000 and convert them to integer numbers.

If you are using the thresholding option of MigClim (i.e. asking it to reclassify continuous projections into binary values: the habitat is suitable or not), then you should simply also multiply the threshold value that you obtained for the [0:1] range by 1000. E.g. if your Maxent threshold was of 0.5, then you would set `rcThreshold=500`.

If you wish to work without the thresholding option, then the values of habitat suitability (in the range 0:1000) are used as a cell "invasibility" index and are interpreted as an absolute probability of presence conditional on the species dispersing to the cell (see also equation 1 in this document). In other words, all other things being equal, a cell with habitat suitability of 600 is twice as likely to be colonized than a cell with habitat suitability of 300. Note that depending on how you calibrated your Maxent model, the predictions are not necessarily absolute probabilities. If they are not, then they you will have either to rescale them, or to use the thresholding option.

What is the modeling unit of MigClim?

The basic modeling unit of MigClim is a cell (or pixel) that is occupied (inhabited) or not. While an

occupied cell can, to some extent, be thought of as a population, MigClim does not model the number of individuals within a cell, nor each individual independently. Therefore, parameter values in the model should reflect values for an entire cell (population), which has both advantages and limitations (discussed in Engler and Guisan 2009).

The colonization of a new cell therefore means that the cell becomes occupied by a new population, though initially this "population" could be seen as composed of only 1 individual. Obviously the potential of an occupied cell to colonize other cells depends on the size of its population (a large population will likely produce more propagules), which is likely to be a function of the time since the cell became colonized (this is naturally a simplification as there are more parameters that affect population size than just time since a cell became colonized). To allow accounting for these issues, a "propagule production potential" parameter [`propaguleProd`] that affects the probability with which a given cell can produce propagules is available.

Can MigClim run simulations without environmental change?

Yes, it can. Implementing environmental change is not a requirement in MigClim. In fact, simulations without environmental change are even easier to implement because you don't need to have different habitat suitability maps but just one. An example of simulations without environmental change could be if you want to model the spread of an invasive species trough the landscape. The initial distribution map could be the point of introduction of the invasive species, and the habitat suitability map the potential suitable habitat for the species. To run a simulation without environmental change, simply set [`envChgSteps=1`].

Part 2: The MigClim model explained

Basic principle of the MigClim model

The basic principle of the MigClim model is the following: The user gives an initial species distribution (the starting point of the simulation) and one or more habitat suitability maps (maps indicating which cells are suitable for the species and which are not, at one or several points in time). Using this data and the information about the dispersal ability of the species, MigClim will simulate the dispersal of the species and produce a potential distribution map that accounts for dispersal limitation.

The basic time unit in MigClim is a "dispersal step" ([`dispSteps`] parameter), and corresponds to one dispersal event (one loop in the flow chart shown in Fig. 2). During a dispersal step, each cell that is occupied and able to reproduce is given the opportunity to colonize new, empty and suitable, cells. Each occupied cell will also increase its "age" by 1 time unit. For instance, if a cell has been colonized since 4 dispersal steps, it will have an age of 4. In practice, a dispersal step will often be equal to one year, as most organisms disperse once a year or can be modeled as such.

Another concept that is important in MigClim is that of "environmental change step" ([`envChgSteps`] parameter). To understand it, one should know that MigClim was initially developed to refine projections of species distributions under climate change scenarios. An environmental change step corresponds to a change/update in the habitat suitability map that is used in MigClim to define which cells can be colonized and which cannot (or the probability with which they can get colonized).

The way that dispersal step [`dispSteps`] and environmental change step [`envChgSteps`] relate is that the dispersal step loop is nested within the environmental change step loop (see the model's flow-chart in Fig. 2). The total number of dispersal events that will be performed within a simulation is thus equal to [`envChgSteps`] × [`dispSteps`].

In practice, it is generally convenient to set the number of environmental change steps and the number of dispersal steps so that each dispersal step corresponds to one year (because most organisms populations can be modeled as dispersing once a year). Here is an example to illustrate what is meant: Let's assume that we want to model the potential distribution of a specie under a climate change scenario for the year 2100 and that we know its initial distribution at the year 2000. How do we proceed ?

A first option could be to simply model its potential distribution by 2100 and then simulate 100 dispersal steps, one for each year from 2001 to 2100. In this case, we would set [`envChgSteps=1`] because we have only one habitat suitability map (i.e. the one giving the habitat suitability by the year 2100), and [`dispSteps=100`], because we want to run the simulation for 100 years. So, if we multiply: [`envChgSteps`] × [`dispSteps`] = 1 × 100 = 100 years, which is precisely what we wanted to do. Nice.

While this option seems easy to implement, it is by no means an ideal one. The problem here is that we modeled the habitat suitability only for the year 2100, with no intermediate step. Thus, it is very possible that, by doing so, we create important gaps between the initial distribution of the species and its potential distribution by 2100. In fact, the gaps that we would create might well be beyond the species' dispersal abilities and the model will thus tell us that the species will go extinct because it is unable to colonize any of the newly suitable habitat.

A better way to proceed is therefore to model the change in habitat suitability for smaller time intervals. For instance, we could decide to model the change in habitat suitability every 5 years. This means that, from 2001-2100, we have to update the habitat suitability 20 times (100/5 = 20). The number of environmental change steps must thus be set to [`envChgSteps=20`]. Since we choose to update the habitat suitability every 5 years, this means that we must set [`dispSteps=5`], so that each dispersal step corresponds to one year. If we multiply: [`envChgSteps`] × [`dispSteps`] = 20 × 5 = 100 years, which, again, is what we wanted to achieve. This time however, our simulation will likely be more correct since we implemented climate change as a series of 20 steps of small magnitude each, rather than just one step of big magnitude as in the first method discussed above.

As an extreme case, we could also choose to set [envChgSteps=100] and [dispSteps=1]. This means that we would update habitat suitability every year, and thus run only one dispersal event per habitat suitability map update. This would require to produce 100 habitat suitability maps and might not be ideal if there is an important variability from one year to the other (i.e. climate change projections contain uncertainty, and it might thus be better to compute average values over a few years).

Dispersal simulation flow in MIGCLIM

Dispersal is simulated through a number of decisions that are taken, for each cell (target cell), during each dispersal step (see also flowchart in Fig. 2):

1. Does the target cell represent a suitable habitat? Is it unoccupied?

2. If point 1 is answered positively, the number *n* of source cells within the dispersal Kernel's [dispKernel] maximum dispersal distance is computed. Source cells are occupied cells that can act as propagule sources to colonise a target cell. Optionally, a barrier layer can be given to prevent dispersal through those cells being part of the barrier. If a barrier cell is found between the target and a source cell, the source cell is ignored.

3. If $n > 0$, the target cell becomes colonised with the combined probability P_{Col} :

$$P_{Col} = \left(1 - \prod_{i=1}^{n} \left(1 - P_{Disp_i} \times P_{\mathrm{Prop}_i}\right)\right) \times P_{Inv} \qquad \text{(eq.1)}$$

   Where $P_{Disp\ i}$ is a probability function of the distance between the target cell and source cell *i* (values of P_{Disp} are entered through the [dispKernel] parameter) and reflects the fact that colonisation probability decreases over distance. $P_{Prop\ i}$ is a probability that is function of the time since the source cell *i* became occupied and represents the propagule production potential of source cell *i* over time (values of P_{Prop} are entered via the [propaguleProd] parameter).
   P_{Prop} can be used to represent time for individuals to reach reproductive maturity and, more globally, the increase of a population's reproductive potential due to an increase in the number of individual plants within a cell over time. P_{Disp} and P_{Prop} are implemented as discrete functions and can easily be modified to fit any shape of seed dispersal curve and increase of reproductive potential over time. Finally, P_{Inv} represents the habitat invasibility of the target cell and generally depends on its suitability for the species. Note that this value can simply be set to 1 if no data is available to calibrate this parameter.

4. Optionally, long distance dispersal (LDD) events can be added to the simulation. LDD events are generated from source cells with a probability [lddFreq] × P_{Prop} in a random direction and at a random distance within a user-defined range. If the cell reached by the long distance dispersing propagule is potentially suitable (satisfying point 1), it becomes colonised. LDD events are not affected by barriers.

5. Steps 1 to 4 are repeated a number of times equal to the number of dispersal steps [dispSteps], which is typically set so that each dispersal step corresponds to one year.

6. Cells that are no longer suitable due to changes in environmental conditions have their values reset to zero. Cells that become unsuitable are reset only after the dispersion stage occurred (steps 1 to 5), because it is assumed that the change of a habitat from suitable to unsuitable is not a discrete but a continuous process. Thus, organisms inhabiting a cell still have the potential to disperse during the environmental change step when the cell turns unsuitable.

7. Steps 1 to 6 are repeated a number of times equal to the number of environmental change steps [envChgSteps]. In each repetition, the habitat suitability is updated to reflect environmental change (e.g. climate change). Simulations without environmental change can be performed by setting envChgSteps=1.

Important: since the modelling unit in MIGCLIM is a cell that is occupied or not, parameter values must represent values for a cell (which can be thought off as a population), not for a single individual.

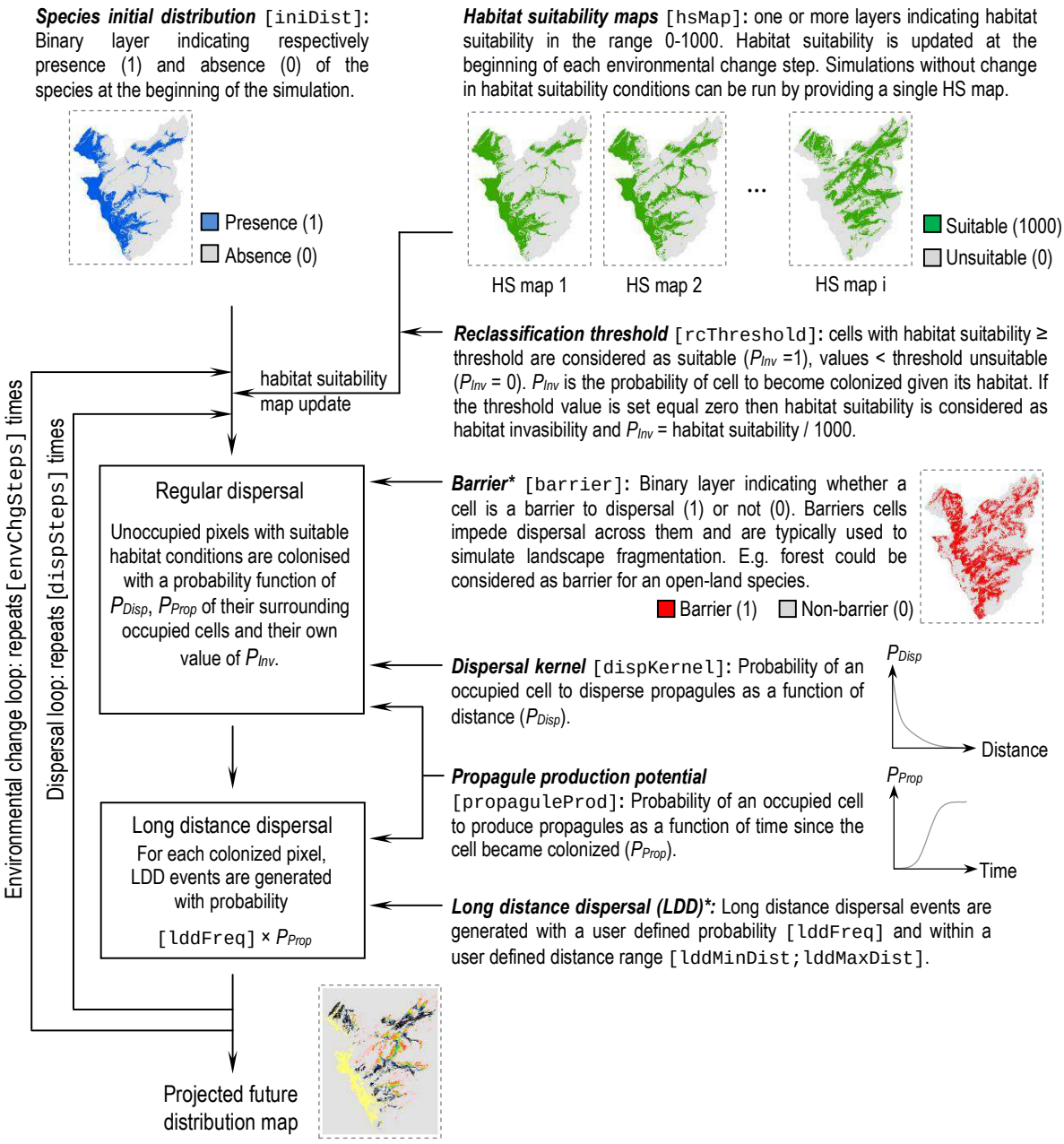


Figure 2. Flow-chart of a dispersal simulation using MIGCLIM R package. [`Parameter names`] refer to the parameters in the `MigClim.migrate()` function.

MigClim parameters detailed description

In order to simulate the dispersal of a species, the MigClim model has a number of compulsory and optional parameters. All of these parameters can be modified so that their value best suits the species one wishes to model. Here is a list of the different MigClim parameters and their description (the order in which the parameters are listed corresponds to their order in the `MigClim.migrate()` function). Note that when in R you can always obtain help for a function by typing "?" followed by the name of the function, e.g. "`?MigClim.migrate`" (the explanations in this user guide are however more in-depth than what is found in the R help files).

Here are the default values of the `MigClim.migrate()` function:

```
MigClim.migrate(iniDist="InitialDist", hsMap="HSmap", rcThreshold=0, envChgSteps=1,
dispSteps=1, dispKernel=c(1.0,1.0), barrier="", barrierType="strong", iniMatAge=1,
propaguleProd=c(1.0), lddFreq=0.0, lddMinDist=NULL, lddMaxDist=NULL,
simulName="MigClimTest", replicateNb=1, overWrite=FALSE, testMode=FALSE,
fullOutput=FALSE, keepTempFiles=FALSE)
```

[iniDist]

This first parameter's name stands for "initial distribution", and, unsurprisingly, is the one to which you need to pass the initial distribution of your species. The species' initial distribution is a spatial parameter: it's a map that indicates for each cell of the study area whether it contains the species or not at the beginning of the simulation. This layer should contain only two types of values: 0 (species is absence) or 1 (species is present). These values must be entered as integer values, floating point values are not accepted.

The value of [`iniDist`] can be entered either as a data frame, or as a raster:

Data frame format:

When entered as a data frame, then [`iniDist`] must be a data frame (or a matrix) with exactly 3 columns (in this order): X coordinate, Y coordinate and value of the initial distribution (either 0 or 1). Here is an example of how this input looks like: `iniDist=MigClim.testData[,1:3]` (a data frame with 3 columns).

Raster format:

To enter the species' initial distribution in a raster format, the value of [`iniDist`] must be set to the name of the raster file (which must be saved somewhere on disk). In other words, the value of [`iniDist`] is a string – the name of a raster file – and not a raster object itself. The following four raster formats are supported: ascii grid (.asc), GeoTIFF (.tif), R raster (no file extension) and ESRI grid (no file extension). These formats offer good compatibility with existing species distribution modeling packages and GIS software.

The name of the file must be given relative to the current R workspace. Here are some examples: Let's assume that we have an ascii grid file called "InitialDistribution.asc" that contains our initial distribution. If this file is located in the current R working directory, then we would set the value of `iniDist="InitialDistribution"` (notice that the value of `iniDist` is a string). If this file is located in a sub-directory called "Species1" of my workspace, then we would set the value of `iniDist="Species1/InitialDistribution"` (again, notice that the value is a string, not a raster).

Note: If you give your input in raster format, then your layer can also contain "NoData" values for locations that are outside of your study area. NoData generally have a value of -9999.

Important: While you can chose to enter your data in either data frame or raster format, you must use the same method for all spatial parameters inputs (there are 3 spatial parameters in the `MigClim.migrate` function: [`iniDist`], [`hsMap`], [`barrier`]).

[hsMap]

In order to simulate the dispersal of a species through the landscape, MigClim needs to know which cells are favorable for the species to establish and which are not. The way that MigClim get this information from you is through "habitat suitability maps", which are entered through the [`hsMap`] parameter of the `MigClim.migrate()` function.

Depending on whether you want to implement change in environmental conditions during your simulation, this input will consist in one or several spatial layers (maps). Each habitat suitability map must contain integer values in the range [0:1000], where 0 is the lowest value and 1000 the highest value of habitat suitability. This range of values was chosen because it offers direct compatibility with the outputs of the Biomod software which are also in the range [0:1000].

Important: the habitat suitability values must be integer numbers in the range 0 to 1000. Floating point numbers are not accepted. This done to save memory space (i.e. storing integer number eats less space).

As with the [`iniDist`] parameter, and as with all spatial parameters in MigClim, the value of [`hsMap`] can be given either as a data frame, or as a name of a raster file.

Data frame format:

When entered as a data frame, [`hsMap`] must be a data frame (or a matrix) with a number of columns equal to the number of habitat suitability maps that you wish to enter (so at least 1 column). Unlike the [`iniDist`] data frame, this data frame should not contain any X and Y coordinates, because MigClim assumes that the order of the rows in the [`iniDist`] and [`hsMap`] data frames are the same (so make sure that they are in the same order).

In our test simulation, we have 5 different habitat suitability maps, and hence our input will look like this: `hsMap=MigClim.testData[,4:8]` (a data frame with 5 columns – each column representing a habitat suitability map. Note that there are no X and Y coordinates in this data frame).

Raster format:

To enter the habitat suitability maps in raster format, the value of [`hsMap`] must be set to the base name of the raster file. What is meant by "base name", is that all of the habitat suitability raster files should have the same "base name" followed by a number that indicates the order in which they should be used (the numbering must start with 1). For instance, assuming that we have 5 habitat suitability maps, and that the base name of our habitat suitability maps is "SuitabilityMap", then the five files must be named "SuitabilityMap1.asc", "SuitabilityMap2.asc", "SuitabilityMap3.asc", "SuitabilityMap4.asc" and "SuitabilityMap5.asc" (note that I have used the ".asc" extension here, but this would change if you use a raster format different from ascii grid).

The value passed to the [`hsMap`] parameter is then the base name of the habitat suitability files, e.g. `hsMap="SuitabilityMap"`. If the files are not located in the current R working directory, then you have to indicate the relative path to this directory in the input, for instance `hsMap="Species1/SuitabilityMap"`.

One question that you should have at this stage is "since I only indicate the base name of the habitat suitability maps, how does the damn thing know how many habitat suitability maps I have?". Well, that is what the [`envChgSteps`] parameter is there for (explained later in this tutorial, stay tuned).

The following four raster formats are supported: ascii grid (.asc), GeoTIFF (.tif), R raster (no file extension) and ESRI grid (no file extension). These formats offer good compatibility with existing species distribution modeling packages and GIS software.

Important: All of the raster files have to be located in the same directory.

Important: All of the raster files must have the exact same base name and be followed by a number – starting with 1 – indicating the order in which they should be used. The first file must thus always end in "1". The numbers of the habitat suitability files must be consecutive (no gaps in the numbering – thank you very much).

Important: All spatial inputs (`[iniDist]`, `[hsMap]`, `[barrier]`) *must have exactly the same extent, cell size and extent (number of cells per row and column). Should any of your spatial data not satisfy this criteria, your results will be wrong.*

[rcThreshold]

MIGCLIM offers two options to handle the habitat suitability data entered via the [`hsMap`] parameter: binary mode and continuous mode. [`rcThreshold`] is the parameter that allows switching to either of these modes.

Important: the values of [`rcThreshold`] *must be an integer number between 0 and 1000. Floating point numbers are not accepted.*

In "binary" mode, the input habitat suitability data (which as you may remember must be given in the range [0:1000]) are converted into binary data: the habitat is either fully suitable (1000) or not (0). The reclassification is done via the [`rcThreshold`] parameter. Values of habitat suitability >= `rcThreshold` are reclassified as suitable habitat (1000), while values < `rcThreshold` are reclassified as unsuitable habitat (0). Thus, to use the "binary" mode, the value of [`rcThreshold`] must be in the range 1 to 1000.

"continuous" mode is entered when setting the value of [`rcThreshold=0`]. In that case, the habitat suitability data that is entered via the [`hsMap`] paramter will not be converted into binary values. Instead, the values of habitat suitability will be used as a conditional probability that a cell becomes colonized (the value of P_{Inv} that you can see in equation 1 of this document). In "continuous" mode, the habitat suitability value is thus interpreted as a value of "habitat invasibility". The actual value of invasibility (P_{Inv}) is computed simply by dividing the habitat suitability value of a given cell by 1000. So, for instance, a value of 1000 corresponds to a probability of 1, and a value of 395 to a probability of 0.395. It is important to understand that in "continuous" mode, the values of habitat suitability are interpreted as an absolute probability of presence conditional on the species dispersing to the cell (see equation 1 in this document). In other words, all other things being equal, a cell with habitat suitability of 600 is twice as likely to be colonized than a cell with habitat suitability of 300. Note that depending on how you calibrated your model, the predictions are not necessarily absolute probabilities. If they are not, then they you will have either to rescale them, or to use the thresholding option (i.e. "binary" mode).

Note: when using "binary" mode, MIGCLIM will simply reclassify you habitat suitability values into either 0 or 1000. Thus, if you already have binary data as input, you can simply reclassify these data into either 0 or 1000 by yourself and then use this as MIGCLIM input (in this case you can set [`rcThreshold`] *to either 0 or any value between 1-1000 the result will be the same).*

[envChgSteps]

The [envChgSteps] parameter is where you indicate the number of "environmental change steps" that should be performed during your simulation. The number of environmental change steps corresponds to the number of times the habitat suitability data should be updated during the simulation (and must thus correspond to the number of habitat suitability maps that you have). If you wish to run a simulation without any modification in the habitat suitability map (e.g. if you're modeling the spread of an invasive species), then simply set [envChgSteps=1]. If you wish to update your habitat suitability map 5 times, as in our example, then set [envChgSteps=5].

Important: for each environmental change step, [dispSteps] dispersal steps are run.

Note: The minimum value for [envChgSteps] *is 1, the maximum value is 295.*

Note: If you have entered your [hsMap] *data in a data frame format, then the* [envChgSteps] *parameter may seem a bit redundant since the number of columns of your data frame already defines the number of environmental change steps (but you still have to enter it!). However, if you enter your* [hsMap] *data in raster format (indicating the "base name" of your habitat suitability files), then you can see why the* [envChgSteps] *parameter is important: without it there is no way for the function to know how many times you want to update your habitat suitability map during the simulation.*

[dispSteps]

The [dispSteps] parameter is where you indicate how many dispersal steps you want to perform.

Important: remember that the dispersal step loop is nested within the environmental change loop (see the figure of the MIGCLIM simulation flow earlier in this document). Thus, [dispSteps] *dispersal steps are run for each environmental change step.*

In our tutorial example, the interval between two successive habitat suitability maps is of 5 years. Since we assume that our species disperses once a year (hence 1 dispersal step = 1 year), we have to set [dispSteps=5]. The total number of dispersal steps that will occur during our simulation is thus of [envChgSteps=5] x [dispSteps=5] = 25.

Note: The minimum value for [dispSteps] *is 1, the maximum value is 99.*

[dispKernel]

What is important to understand at this stage is that MIGCLIM considers two kind of dispersal processes: "short distance dispersal", which is the normal dispersal mode and "long distance dispersal" (abbreviated LDD), which represents rare events where seeds are dispersed by non-usual means and travel longer distances than usual. To illustrate this with a real-world example, think about an anemochorous species, i.e. a species dispersing its seeds by wind. The short dispersal distance for this species represents the usual distance that seeds are dispersed under normal wind conditions, say for instance 50 meters. Long distance dispersal on the other hand corresponds to rare events where a seed will be dispersed over a much longer distance, say up to 1000 meters or 10 kilometers for instance, through an exceptional strong wind or if it gets attached to an animal that will transport it.

For computing efficiency reasons, the approach taken in MIGCLIM is to simulate these two kind of dispersal as separate processes. Short distance dispersal is entered via the dispersal kernel [`dispKernel`] parameter, which is the focus of the present section. Long distance dispersal distance and frequency is set via three parameters: [`lddFreq`], [`lddMinDist`] and [`lddMaxDist`], that are discussed later in this document.

A dispersal kernel is a function that indicates the probability of a propagule to disperse at a given distance from its source. Because the MIGCLIM model is a cellular automaton (i.e. a model based on a grid of cells – the pixels of your input rasters), the values for the dispersal kernel parameter [`dispKernel`] should also be expressed in cell units. In other words, the [`dispKernel`] parameter is a vector that gives the probability of a cell to disperse propagules *(P_{Disp})* to a distance of 1 cell, 2 cells, 3 cells, etc... until the maximum dispersal distance of your species is reached.

Figure 3 below illustrates a 5 cell dispersal kernel. If the cell size is of 20 meters, then this corresponds to a maximum dispersal distance of 5 x 20 = 100 meters. Fig. 3a shows the spatial distribution of the dispersal kernel: the central cell represents the propagule source, and the different grey patterns indicate the respective distance classes to which each cell belongs. The colors of the cells match with the colors of the bar plot values shown to the right (Fig. 3b and 3c). Distance values that are non-integer numbers (e.g. diagonals) are rounded to their closest integer number and attributed to that distance class. E.g. the d1 and d2 arrows indicate cells that belong to distance class 2 and 3 respectively. Fig. 3b gives an example of a negative exponential kernel (this kernel is used in the tutorial section presented further down in this user guide). P_{Disp} represents the probability for a source cell to disperse a propagule at a certain distance (see also equation 1 of this document). Each distance class is represented by a different pattern of grey that matches with those found in Fig. 3a. Fig. 3c shows an example of a kernel where all distance classes have an equal value of P_{Disp} = 1. Using this dispersal kernel, any cell within a distance ≤ 5 cells is colonized with certainty (provided other conditions are fulfilled).

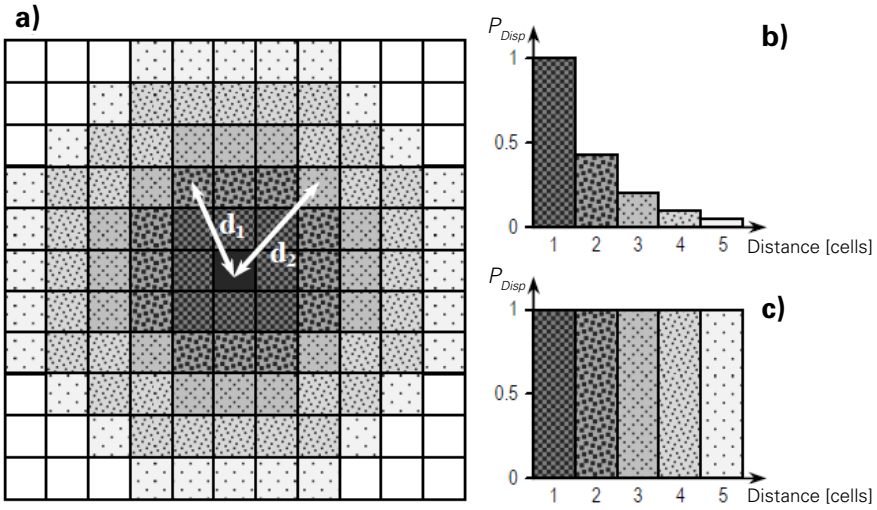


Figure 3.
illustration of two different 5 cell dispersal kernels.

The dispersal kernel of Fig. 3b would be entered as: `dispKernel=c(1.0,0.4,0.16,0.06,0.03)`
The dispersal kernel of Fig. 3c would be entered as: `dispKernel=c(1,1,1,1,1)`

Important: dispersal kernel values must be entered as a vector of probabilities. Each probability corresponds to a distance class measured in cell units. The probability values must be in the range [0:1] (where 0 means P_{Disp} = 0% and 1 means P_{Disp} = 100%).

Note: in its simplest form, a dispersal kernel can be given as having a constant value of 1 over the entire dispersal distance (as shown in Fig. 3c). This might not be fully realistic as we know that the value of P_{Disp} generally decreases with distance from the source cell, but if you have no data to calibrate the dispersal kernel of a species this might be an option.

[barrier] and [barrierType]

In MigClim, the optional [barrier] and [barrierType] parameters can be used to indicate cells across which dispersal cannot occur. Barrier cells are considered as permanently unsuitable (i.e. they cannot become colonized), but unlike regular unsuitable cells, the also impede dispersal across them (see figure below). In example, for a deer species, both a parking lot and a highway are unsuitable habitats, however, while the deer can disperse through a parking lot, it might not be able to cross a highway.

If you do not wish to implement barriers to dispersal into your simulation, simply set [barrier=""].

The [barrier] parameter must be given as a layer of binary and integer values: 1 (cell is a barrier) and 0 (cell is not a barrier). Like all spatial parameters (i.e., [iniDist], [hsMap] and [barrier]), it can be entered either in a data frame/matrix/vector format (in this case a data frame with a single column), or as a raster file in one of the following formats: ascii grid (.asc), R raster (no extension), geoTIFF (.tif) or ESRI grid (no extension). Note that if you want to enter the value of the [barrier] parameter as a raster file, you should indicate the name of the file relatively to the current R working directory, e.g., barrier="Speices1/Barrier". In other words, the input value of the parameter is a string, not a raster object (this is the same as for the [iniDist] and [hsMap] parameters).

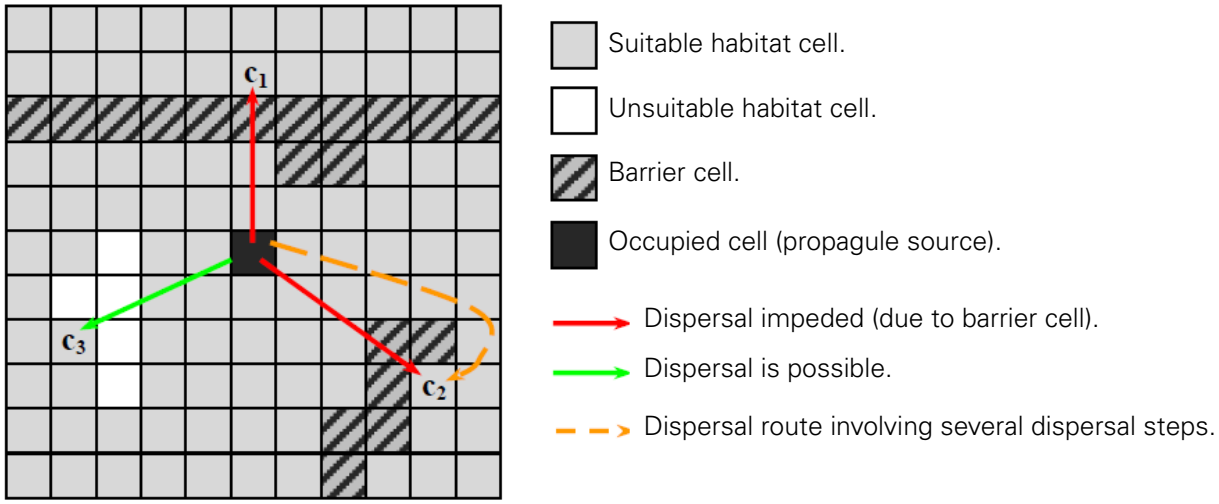


Figure 4. Illustration of the difference between barrier cells and unsuitable habitat cells.

Figure 4 illustrates the difference between barriers cells and unsuitable cell. The example assume that the only cell occupied by our species is the central cell (in black) and that it can disperse propagules to a distance up to 10 cells (which means that all cells of our small landscape are within reach and thus that dispersal distance is not a limiting factor here). Our landscape also contains unsuitable cells (white colored) and barrier cells (striped).

The cells indicated as "c_1" and "c_2" are examples of cells that cannot be colonized due to the presence of barrier cells (a straight line between the black source cell and these cells crosses a barrier cell, as

illustrated by the red arrows). To the contrary, the "c_3" cell can be colonized, because the unsuitable cells (in white) that are located between "c_3" and the source cell (in black) do not impede dispersal across them (illustrated by the green arrow).

To summarize, unsuitable cells cannot host the species (and therefore cannot be colonized), but dispersal across them is possible. Barrier cells too are unsuitable for the species (and thus cannot be colonized), but on top of that they also impede dispersal across them.

Note that propagule dispersal can only occur in a straight line, and that barrier cells cannot be circumvented in a single dispersal step. For instance, the "c_2" cell in Fig. 4 will eventually become colonized (orange dashed arrow), but this will involve several dispersal steps. In other words, the source cell cannot disperse directly to "c_2" by circumventing a barrier cell, even if the actual distance (including the detour involved in the circumvention) is smaller than the dispersal distance.

Two kinds of barriers can be implemented in MigClim: "`strong`" barriers and "`weak`" barriers. This is entered in the `MigClim.migrate()` function via the [`barrierType`"] parameter (`barrierType="strong"` or `barrierType="weak"`). Essentially, the difference is that strong barriers are more restrictive than weak barriers. Note that a barrier layer is entirely "strong" or entirely "weak" (you cannot have a mix of strong and weak barriers).

To determine whether there is a barrier between a source cell (black cell in Fig. 5) and a sink cell (orange cell in Fig. 5), the algorithm looks whether 5 different straight line paths (represented by arrows in Fig. 5) intersect a barrier cell or not. As illustrated in the figure, the 5 paths connect each corner as well as the center of the source and sink cells.

→ When setting `barrierType="strong"`, dispersal is impeded between the two cells when more than 1 of these paths intersects with a barrier cell.

→ When setting `barrierType="weak"`, then dispersal is only impeded when all of the 5 paths intersect with a barrier cell.

Fig. 5 shows two examples where using the "`strong`" barrier setting would impede dispersal between the two cells, while using the "`weak`" setting would not.

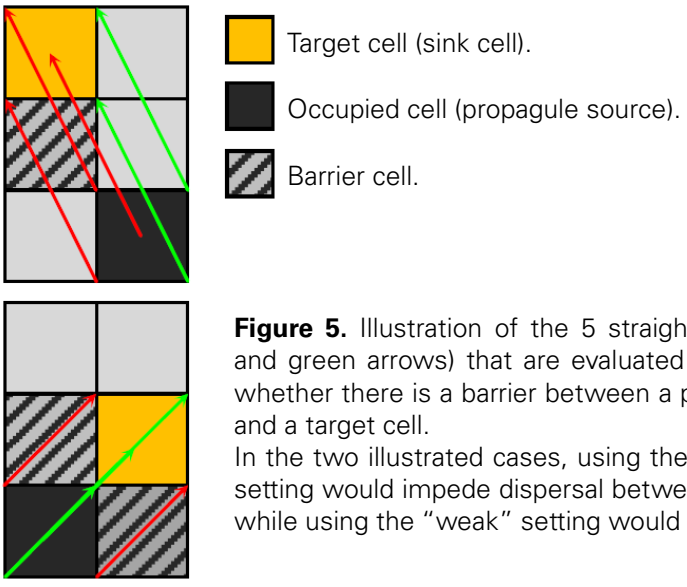


Figure 5. Illustration of the 5 straight line paths (red and green arrows) that are evaluated when searching whether there is a barrier between a propagule source and a target cell.
In the two illustrated cases, using the "strong" barrier setting would impede dispersal between the two cells, while using the "weak" setting would not.

Important: Long distance dispersal events are not affected by barrier features. Barriers only affect regular (non-LDD) dispersal.

Note: barrier cells remain barriers during the entire simulation and cannot be updated during the simulation.

[lddFreq], [lddMinDist] and [lddMaxDist]

Long distance dispersal (LDD) events are randomly generated with a user-defined frequency [lddFreq] within a used-defined distance range [lddMinDist; lddMaxDist]. The frequency of LDD events is also modulated through the propagule production potential of the considered cell: during each dispersal step, the probability for an occupied cell to produce a long distance dispersal event is of [lddFreq] x P_{Prop}.

LDD events aim at representing non-standard ways of propagule dispersal. E.g., a seed from a myrmecochorus species can occasionally be dispersed by another animal over much larger distances. Note that LDD events are not affect by barriers to dispersal.

If you do not wish to implement LDD into you simulation, simply set [lddFreq=0] (in which case [lddMinDist] and [lddMaxDist] can be set to anything – in fact you don't even need to add them to the function call).

If you choose to implement LDD, then [lddFreq] must be a value in the range]0:1] indicating the probability with which an occupied cell that has reach its full propagule production potential (P_{Prop} = 1) will produce a LDD event at each dispersal step. For instance, if you set [lddFreq=0.01], then, on average, 1 occupied cell (with full propagule production potential) in 100 will generate a LDD event. If a given cell has not reached its full propagule production potential, then the probability with which it will generate a LDD event at each dispersal step is equal to [lddFreq] x P_{prop} (see equation 1).

The distance at which LDD events should be generated must be indicated through the LDD minimum distance [lddMinDist] and the LDD maximum distance [lddMaxDist] parameters. The distance unit of these parameters is a "cell", so you have to set these values depending on the spatial resolution at which you work (the cell size of your raster data). For instance, if you set [lddMinDist=6], [lddMaxDist=50] and your cell size is of 20 meters, then LDD events will be randomly generated within a distance range of 120 to 1000 meters. Note that the value of [lddMinDist] must be larger than the largest distance of the dispersal kernel [dispKernel]. For instance it the dispersal kernel is a vector of length 5, then [lddMinDist] must be >= 6. The value of [lddMaxDist] must also be larger than the value of [lddMinDist]. Also, both [lddMinDist] and [lddMaxDist] must be integer numbers.

> *Note: the probability of an LDD event to reach any cell within the defined distance range [lddMinDist – lddMaxDist] is independent of that distance. The probability remains constant across the entire distance range.*
>
> *Important: Long distance dispersal events are not affected by barrier features. Barriers only affect regular (non-LDD) dispersal.*

[propaguleProd] and [iniMatAge]

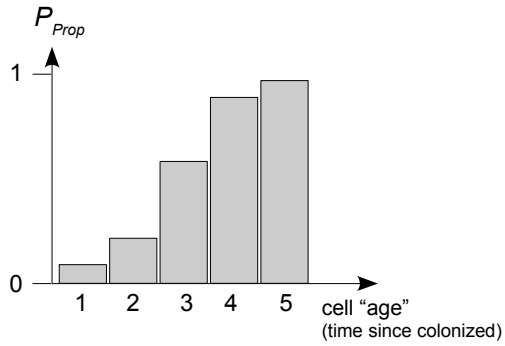
The probability of a source cell to produce propagules (P_{Prop}) as a function of time since the cell became colonized is specified via 2 parameters: initial maturity age [iniMatAge] and a vector [propaguleProd] indicating the probability of propagule production (P_{Prop}) for each age between the initial maturity age and full maturity. This parameter can be used as a proxy for population growth in the cell, or for instance to reflect that a species might need several years before starting to produce propagules, and even more time to reach its full reproductive potential. The time unit is a dispersal step, which will usually, but not necessarily, represent one year.

The value of [iniMatAge] must be set to the cell "age" at which a cell will start to produce propagules (i.e. the age when P_{Prop} becomes larger than 0). Remember that in MIGCLIM the time unit is a dispersal step, and hence, cell "age" is measured in dispersal steps.

The value of [propaguleProd] must be a vector indicating the probability P_{Prop} associated to each cell age, from its initial maturity age [iniMatAge] to one year before full maturity. When a cell has reached its full maturity age (P_{Prop}=1), it is assumed that its value of P_{Prop} will remain equal to 1 indefinitely, and this is

why you don't need to specify it anymore. Note that values of [propaguleProd] should be given in the range [0:1], where 0 means P_{Prop} = 0% and 1 means P_{Prop} = 100%.

To give you an idea, here are two examples of how the increase in propagule production potential over time might look:

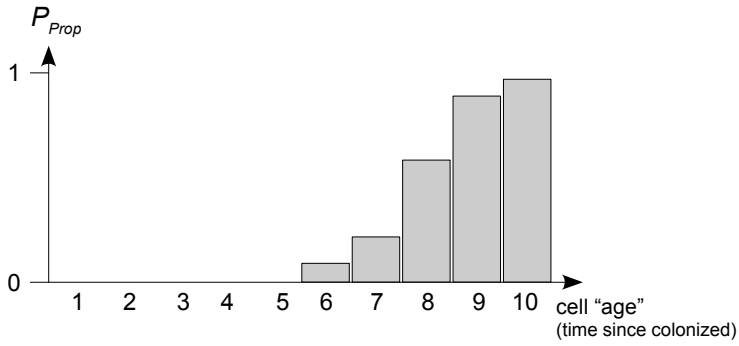


Example 1. Here the species starts to have some propagule production potential since the first dispersal step after the pixel became colonized (i.e. pixel age = 1). The value of P_{Prop} then increases in a sigmoid fashion to reach its maximum at the age of 5. From the age of 5 on, the value of P_{Prop} will always remain at its maximum of 1.

These values would be entered as follows:

```
iniMatAge=1
propaguleProdProb=c(0.02, 0.1,0.5,0.9)
```

Note that we do not need to indicate the value of pixel age 5 because this value is 1.



Example 2. In this case, the species will not be able to reproduce before it has reached an age of 6 dispersal steps (in general 1 dispersal step = 1 year).
From the age of 6, the value of P_{Prop} then increases in a sigmoid fashion to reach its maximum at the age of 10. From the age of 10 on, the value of P_{Prop} will always remain at its maximum of 1.

These values would be entered as follows:

```
iniMatAge=6
propaguleProdProb=c(0.02, 0.1,0.5,0.9)
```

Note that we do not need to indicate the value of P_{Prop} for pixel age 10 because this value is 1.

As can be seen from the examples above, the length of the [propaguleProd] vector is always equal to the full maturity age - initial maturity age: For instance, in example 1, the length of the [propaguleProd] vector is of 4 (5 – 1 = 4), and the same for example 2 (10 – 6 = 4).

The only exception to this is when the initial maturity age and the full maturity age are the same (i.e. the propagule production potential increases from 0 to 1 within one dispersal step). In this case the value of [propaguleProd] must be set to `propaguleProd=c(1.0)`. E.g. if you want your cells to reach full propagule potential in the first dispersal step after they became colonized, you would enter this as: `iniMatAge=1, propaguleProd=c(1.0)`.

Important: Once cells have reached their full maturity, their value of P_{Prop} remains always equal to 1 (unless they become decolonized, obviously) and has thus no longer to be specified.

Important: In MigClim, time is measured in dispersal steps units. Usually one dispersal step should be set so that it equals one year (this assumes that the species will disperse once a year).

Note: The cells that are occupied at the beginning of a simulation (i.e. the species initial distribution) are set to have full propagule production potential (P_{Prop} = 1).

[simulName]

The [simulName] parameter is where you can indicate the "base name" to be used for all of the simulation's outputs. The value of this parameter must thus be a text string. For instance, in the tutorial example presented later in this document, we want to use "MigClimTest" as our based name, and hence we set simulName="MigClimTest". As a result, all of our outputs will be named as "MigClimTest" + something.

The value of [simulName] is also used to name the directory in which the all outputs from the simulation will be saved. In the case of our tutorial simulation, the outputs will thus be saved in a folder named "MigClimTest". This folder will be created in the current R workspace, so make sure to correctly set the R workspace before starting your simulation.

[replicateNb]

The idea behind this parameter is simple: since MIGCLIM simulations generally involve some randomness (this actually depends on the parameter values that you have entered), it can be interesting to repeat several times a same simulation and look at the variability due to random processes. The number of times a simulation should be replicated is entered via the [replicateNb] parameter. It must be an integer number >= 1. For instance, in the tutorial example we wish to replicate our simulation 3 times, hence we set [replicateNb=3]. If you wish to run your simulation only once (i.e., no replication), set [replicateNb=1].

Note that when the value of [replicateNb] is > 1, the "base name" of the outputs of each replicate becomes [simulName] + number of the replicate. For instance, in the tutorial example presented later in this document, we set simulName="MigClimTest" and replicateNb=3, hence the outputs will be named "MigClimTest1", "MigClimTest2" and "MigClimTest3". When replicates are completed, MIGCLIM will also compute an average value of all replicates and save it into the output folder. The average outputs are also named after the [simulName] but do not have any replicate number in their name.

[overWrite]

If [overWrite=TRUE] then any existing file with the same name as an output of the MigClim.migrate() function will be mercilessly overwritten. If [overWrite=FALSE] then the function will stop if any output file does already exist (this is verified before the simulation starts).

[testMode]

The [testMode] parameter allows you to check your input data without running the actual simulation. If [testMode=TRUE] then the MigClim.migrate() function will check all the provided input data but will not run the actual simulation. If [testMode=FALSE] then the simulation is run.

This can be useful to check your data before running a batch of several simulations, since it will allow to quickly see if any of the planned simulation will crash due to user-input errors.

[fullOutput]

If [fullOutput=TRUE], the current state of the simulation is written to an ASCII grid file after each dispersal step (allowing to spatially reconstruct the dispersal process at each step if needed). If [fullOutput=FALSE] (the default value), only the final state of the simulation is written to an ASCII grid file. Note that in all cases, the numerical values of how many cells are colonized, lost, etc..., are available for each dispersal step in form of a text file. Hence the [fullOutput] parameter only affects the ASCII files (the spatially explicit output of MIGCLIM).

> *Note: be aware that setting* [fullOutput=TRUE] *can produce a large number of ASCII grid files and that these can be fairly large in terms of disk space (it depends of course on how many cells you have in your study area).*

[keepTempFiles] *

When entering spatial parameters [`iniDist`], [`hsMap`], [`barrier`] in data frame format, what happens is that MIGCLIM does internally convert them to ASCII grid files. When setting `keepTempFiles=FALSE` (the default value), then these ASCII grid files (.asc) created from a conversion process in the function will be deleted when the simulation is completed. If you wish to keep the created ASCII grid files then set `keepTempFiles=TRUE`.

While keeping the ASCII grid files is somewhat redundant (since you already have the same information in data frame format), it can sometimes be helpful to track bugs (you can check if the conversion from data frame to ASCII grid was done properly).

Part 3: Hands-on example with a MigClim test simulation

The aim of this section is to illustrate the use of MigClim by going through a step-by-step example that will show how to run a simulation with the test data that comes with the MigClim R package. Note that this is essentially the same example as the one described in [Engler R., Hordijk W. and Guisan A. The MigClim R package – seamless integration of dispersal constraints into projections of species distribution models. *Ecography,* in review.], but with some more details.

Since MigClim was originally designed to simulate dispersal of plant species under environmental change and landscape fragmentation scenarios (e.g. climate change scenarios), the example hereafter will follow such a scenario. It is however also possible to model dispersal without implementing environmental change into the simulation. If you haven't done it so far, I strongly encourage you to read the first part of this user guide (that's only 20 pages ;-). Knowing the basic principles and the various parameters of MigClim will help you better understand what is going on in this example.

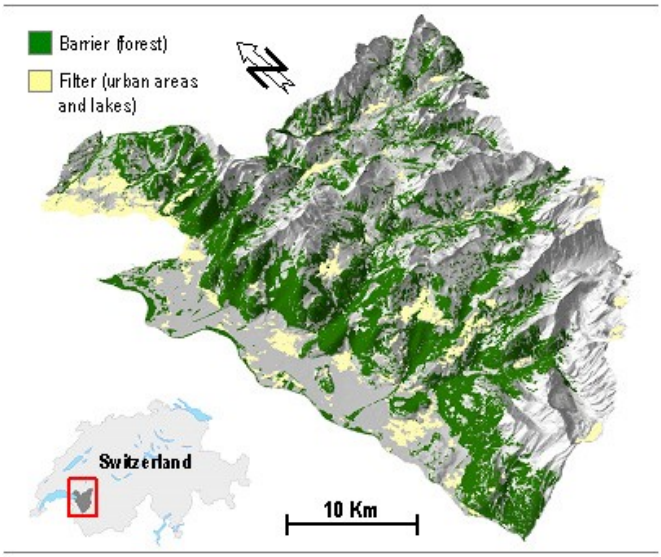


Figure 6. View of the study area (western Swiss Alps) from where the test data originates.In our test example, our aim will be to model the dispersal of a species under a climate change scenario over a period of 25 years, from 2001 to 2025.

Note: Obviously, all the values that we will be using in the tutorial below are only examples and any resemblance with a plant or animal species – alive or extinct – is purely coincidental. In other words, you will have to work out your own parameters when using MigClim for your species – this hurts, I know.

Getting started: Installing R and the MigClim library

If you are interested in using MigClim, chances are that you are already an R guru. If you're new to R however, you should start by going to www.r-project.org and install the R software: it's free, open source and works great. What's not to like?

Next you will want to install the MigClim package. That's real easy to do: in the R console, go to "Packages" > "Install Package(s)". You will be prompted to choose a mirror location near you, after which a (long) list of available packages will be displayed. Scroll down until you see the "MigClim" package, select it and click OK. And voilà, you're up and ready to go. I'm sure you can hardly wait!

Note: MigClim occasionally uses functions from two other R packages: "raster" and "SDMTools". So make sure you are also installing these on your computer (in theory these two packages should install automatically when you instal the MigClim package).

To verify whether a package is installed or not, simply try loading it by typing `library(MigClim)`, `library(raster)`, or `library(SDMTools)`.

Loading and exploring the test data

Before being able to run the test simulation, you will need to load the test data that comes with the MigClim package. And if you haven't done it so far, you should also load the MigClim package itself before that. This is easily done via the following commands:

```
library(MigClim)
data(MigClim.testData,package="MigClim")
```

You should now have a new object, named "`MigClim.testData`", in your R environment. The `MigClim.testData` object is a data frame, let's see what it looks like.

```
head(MigClim.testData)
         X        Y InitialDist hsmap1 hsmap2 hsmap3 hsmap4 hsmap5 Barrier
1 568687.5 115237.5           1   1000   1000   1000   1000   1000       1
2 568887.5 115237.5           1      0      0      0      0      0       0
3 568987.5 115237.5           0      0      0      0      0      0       0
4 568587.5 115337.5           1   1000   1000   1000   1000      0       0
5 568687.5 115337.5           0      0      0      0      0      0       1
6 568787.5 115337.5           0      0      0      0      0      0       1
```

The `MigClim.testData` data frame has 9 columns (how exiting!). It is important to understand that the `MigClim.testData` data frame represents spatial data. Each row of the data frame contains information about a cell of our study area. The two first columns, X and Y, represent the coordinate of a given cell (pixel), while columns 3 to 9 each represent a spatial layer (i.e. a "map"). For instance, the "`InitialDist`" column represents the species initial distribution. Here is a short description of each column:

X and Y: The X and Y coordinates of a cell. Here the coordinates are given in the Swiss coordinate system which is a metric system.

InitialDist: The initial distribution of our species. This column contains only values of either 0 or 1, where 0 indicates the absence of our species, and 1 the presence of our species (cell is occupied). No other values than 0 or 1 are allowed in this input, and the numbers must be integers.

hsmap1 to 5: Each of these 5 columns contains habitat suitability data for a given point in time. "hsMap1" contains the habitat suitability information for the average of 2001-2005, "hsMap2" for the average of 2006-2010, and so on until "hsMap5" that contains the habitat suitability data for the time period from 2021-2025. Note that habitat suitability data must be integer numbers in the range 0:1000.

Barrier: A layer representing barriers to dispersal for our species. Barrier must always be binary layers, where a value of 1 = barrier cell and a value of 0 = non-barrier cell. In the present case, the barrier cells correspond to forested areas as we will assume that our species cannot disperse through forests. No other values than 0 or 1 are allowed in this input, and the numbers must be integers.

Important: It is a requirement for MigClim to work correctly that all the spatial input data (raster datasets) that are used for a given simulation have all exactly the same extent and cell size. In our test data, the spatial resolution of our data is of 100 meters (i.e. pixel size is 100 m).

Important: All values of [`iniDist`], [`hsMap`] *and* [`barrier`] *must be integer numbers. Floating point numbers are not accepted (this is for efficient memory usage reasons).*

Before starting the actual MigClim simulation, we can have a quick look at our input test data. This is not something that is needed to actually run MigClim, so feel free to skip to the next section if you're already bored.

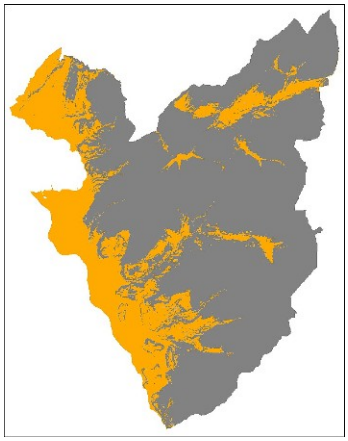
As you probably know, spatial data can be represented using different formats. It can be in a data frame format such as the `MigClim.testData` data frame, but it is also very often in what is called a "raster" format. A raster can be thought of as a two-dimensional array of cells or a two-dimensional matrix. Rasters themselves can be stored in a large variety of formats, such as "ascii grid", "R raster" (the native format used by the R "raster" package), "geoTIFF" or "ESRI grid" (ESRI grid is the native format of the widely used ArcGIS software).

If you want to visualize the data that is given in the `MigClim.testData` data frame, you can convert values from the data frame into raster files. This is easily done using a function from the SDMTools package.

```
library(SDMTools)
dataframe2asc(MigClim.testData[,c(2,1,3:9)])
```

The command above should have generated 7 ascii grid files (they should have a ".asc" extension) in your working directory (the initial distribution, 5 habitat suitability maps and the barrier layer). We can now visualize these raster files either in a GIS or with R, as shown hereafter.

```
InitialDistribution <- raster("InitialDist.asc")
plot(InitialDistribution)
```



Initial distribution of our species.
Orange pixels have a value of 1, indicating they are occupied by the species.
Grey pixels have a value of 0, indicating they are unoccupied.

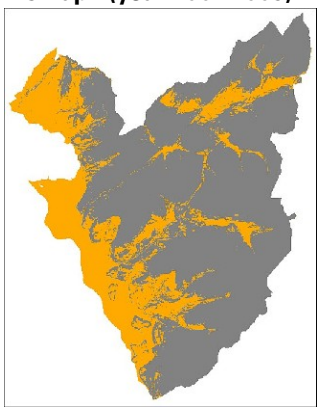
As can be seen from the figure above, our species' initial distribution [iniDist] is a binary map where cells have either a value of 1 (species is present) or a value of 0 (species is absent). Cells outside of the study are have a value of "NoData" (-9999). The initial distribution of our species is mainly restricted to the lowland area to the west of the study area.

> *Important: cell with a "0" value simply indicate that the species is absent from those cells at the beginning of the simulation. However, it doesn't mean that those cells cannot be suitable for the species at the beginning or later on in the simulation. This information is given by the habitat suitability maps* [`hsMap`].
>
> *Note: In this example data, the study area is delimited by "NoData" values (this makes it look nice). However, this is not a requirement and one could also give a value of "0" to all cells outside of the study area.*

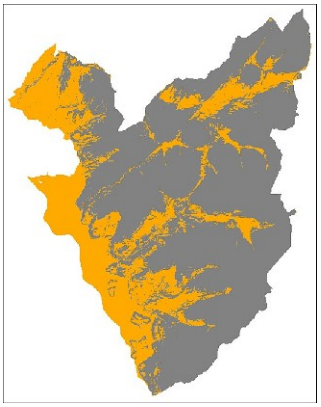
Similarly, we can also have a look at our 5 habitat suitability maps:

```
hsMap1 <- raster("hsmap1.asc")
plot(hsMap1)
```

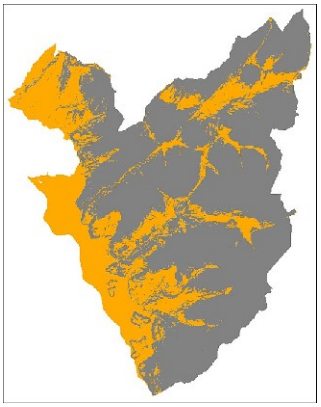
hsmap1 (year 2001-2005)



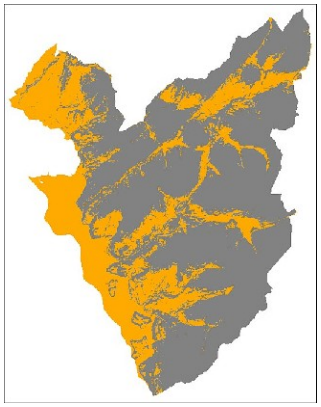
hsmap2 (year 206-2010)



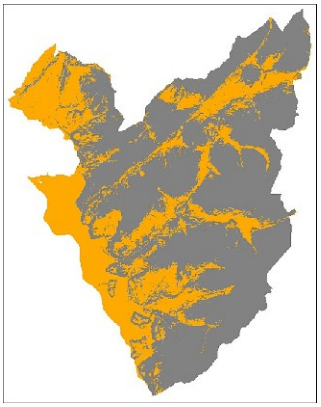
hsmap3 (year 2011-2015)



hsmap4 (year 2016-2020)



hsmap5 (year 2021-2025)

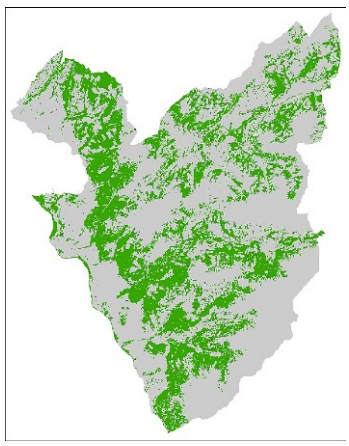


"hsmap1" represent the habitat suitability for the period 2001-2005, "hsmap2" the habitat suitability for the period 2006-2010, … and so on until "hsmap5" that represents habitat suitability for the year 2021-2025. The entire map series thus reflects the evolution of habitat suitability with climate change from 2001 to 2025 for our species (well, I know the change between these 5 maps are not overwhelming, but if you look carefully you will see differences, the suitable habitat is shifting to the east).

Orange cells have a value of 1000 and indicate suitable habitat, which can potentially be colonized/occupied. Grey cells have a value of 0, indicating unsuitable habitat that cannot be colonized/occupied.

Finally we can also display the "barrier" layer (which actually represents forests in the study area):

```
Barrier <- raster("Barrier.asc")
plot(Barrier)
```



Barrier to dispersal layer.
Green pixels have a value of 1, indicating that they are barrier cells (they represent forested areas through which we assume our species cannot disperse).
Grey pixels have a value of 0, indicating they are non-barrier cells.

Running a MigClim simulation

Once you have gathered all your input parameter values, launching a MigClim simulation couldn't be easier. It's done via a single function, `MigClim.migrate()`, to which all parameters are passed. Here is the call to the function that will run our test data:

```
N <- MigClim.migrate(iniDist=MigClim.testData[,1:3],
     hsMap=MigClim.testData[,4:8], rcThreshold=500,
     envChgSteps=5, dispSteps=5, dispKernel=c(1.0,0.4,0.16,0.06,0.03),
     barrier=MigClim.testData[,9], barrierType="strong",
     iniMatAge=1, propaguleProd=c(0.01,0.08,0.5,0.92),
     lddFreq=0.1, lddMinDist=6, lddMaxDist=15,
     simulName="MigClimTest", replicateNb=3, overWrite=TRUE,
     testMode=FALSE, fullOutput=FALSE, keepTempFiles=FALSE)
```

To start the simulation, simply copy the above call to the `MigClim.migrate()` function into your R console, hit your "enter" key and hope for the best. Running this simulation on an laptop that's getting long in the tooth (1.8 GHz Pentium M, 2GB of RAM) took about 42 seconds (your millage might vary depending on the kind of machine you're working on). Note that most of this time (about 40 seconds) was actually spent on converting the data frame inputs ([`iniDist`], [`hsMap`] and [`barrier`]) into ascii grids, and not on the simulation itself. Thus an actual simulation run took less than 1 second (I tried to repeat this simulation 100 times by setting [`replicateNb=100`] and it would take less than 2 minutes). The lesson here is that if you provide the spatial inputs [`iniDist`], [`hsMap`] and [`barrier`] directly as ascii files then the simulation will run a lot faster since if won't have to do the data frame to ascii grid conversion.

Here is a description of the different parameters values used in the `MigClim.migrate()` function call above:

In our test example, we model the dispersal of a species under a climate change scenario over a period of 25 years, from 2001 to 2025. We start with an initial distribution [`iniDist`] for the year 2000 where our species is mainly occupying the lowland habitats located in the western part of the study area. We decide that we want to update our habitat suitability data every 5 years (this update in habitat suitability reflects the climate change projections). Since our simulation runs over 25 years, we need 5 different habitat suitability maps, each reflecting the environmental conditions for a 5-year period (2001-2005, 2006-2010, 2011-2015, 2016-2020, 2021-2025). These maps are entered through the [`hsMap`] parameter (here a data frame with 5 columns), and the number of environmental change steps is entered by setting [`envChgSteps=5`]. The [`rcThreshold`] parameter is here to convert our habitat suitability maps values (in the range 0 – 1000), into binary values indicating whether the habitat is suitable for our species or not. In this example we set [`rcThreshold`] to 500, meaning that values ≥500 will be considered as favorable to our species (such cells can become colonized), while values <500 will be considered unsuitable. We assume that our species can disperse once a year, and hence our total simulation needs to perform 25 dispersal steps (corresponding to the 25 years from 2001 to 2025). As we already set [`envChgSteps=5`], the number of dispersal steps must be set to 25/5 = 5 [`dispSteps=5`]. It is important to keep in mind that for each environmental change step, [`dispSteps`] number of dispersal steps are run (these two nested loops can be seen on Figure 1). The total number of dispersal steps that are simulated is thus equal to [`envChgSteps`] × [`dispSteps`], which in our example equals 25 and corresponds to the 25 years from 2001 to 2025.

An important parameter of any dispersal simulation is the dispersal kernel [`dispKernel`], a vector which indicates the probability of a source cell to disperse propagules as a function of distance measured in cell units. The maximum regular dispersal distance of our species is of 500 m, which corresponds to 5 cells since our input data have a spatial resolution of 100 m (cell size = 100 m). [`dispKernel`] must thus be a vector of 5 values, one for each distance class from 1 to 5 cells. In our example the dispersal kernel follows a negative exponential, with values ranging from 1 for a distance of 1 cell, to 0.03 for a distance

of 5 cells (illustrated in Fig. 3b). We also wish to implement random long distance dispersal events with a frequency of 0.01 [`lddFreq=0.01`], a minimum distance of 6 cells [`lddMinDist=6`] and a maximum distance of 15 cells [`lddMaxDist=15`]. This means that for every cell that has reached full propagule production potential (i.e. $P_{prop} = 1$; see equation 1), a long distance dispersal event will be generated with a probability of 0.01, in a random direction, at a random distance between 6 and 15 cells. If a cell has not reached its full propagule production potential, then the probability with which it will generate a LDD event at each dispersal step is equal to [`lddFreq`] x P_{prop} (see equation 1).

Since we assume that our species is restricted to open habitats and is unable to disperse through forested areas, we want to indicate forested cells as "barriers" to dispersal. We do this via the [`barrier`] parameter: `barrier=MigClim.testData[,9]` (The 9th column of the `MigClim.testData` data frame contains the information about whether a cells contains forested areas or not). The [`barrierType`] parameter can be set either to "`strong`" or "`weak`", here we chose to set [`barrierType="strong"`].

Next we need to indicate how the propagule production potential of newly colonized cells evolves over time. Let's assume our species is annual, and that a newly colonized cell is ready to produce propagules after its first year [`iniMatAge=1`], but will only reach its maximum production potential after 5 years. Note that cell "age" corresponds to the number of dispersal steps elapsed since a cell became colonized, and that in our example one dispersal step is equal to one year. For each age between 1 and 4 years we need to indicate the probability of a cell to produce propagules via the [`propaguleProd`] parameter. In our example, these probabilities are of 0.01, 0.08, 0.5 and 0.92 for ages 1 to 4. We write this as: `propaguleProd=c(0.01,0.08,0.5,0.92)`. Note that we do not indicate the values for age 5 and older, because we want this value to be 1 (and MigClim will assume that the value is always 1 once it has reached the end of the [`propaguleProd`] vector).

Since our simulations includes some level stochasticity, we chose to repeat it 3 times (the average of the 3 repetitions will be automatically computed). We indicate this with [`replicateNb=3`] (note that I chose to repeat the simulation only 3 times for the sake of this example, but for real applications if might be better to increase that number).

The remaining parameters [`fullOutput`], [`simulName`], [`overWrite`], [`testMode`], [`keepTempFiles`] relate to the model outputs and are explained earlier in this document (see here).

In the next section of the tutorial we will look at the different outputs produced by MigClim and see how to interpret them.

Note: In our tutorial example, the interval between two successive habitat suitability maps is of 5 years but you are of course free to use any other interval. For instance, you could implement a change in habitat suitability every year, every 10 years or every 20 years. It all comes down to what kind of data you have available and what kind of values you think does make sense: e.g. implementing climate change every year might not be ideal because annual variability cannot be easily predicted and hence it might be better to work with averages over a few years. On the other hand, implementing climate change only every 20 years might not be optimal either because it could result in changes that are too abrupt between two successive habitat suitability maps.

Note: In the call to `MigClim.migrate()` *illustrated above, I have chosen to enter the spatial parameters values (*[`iniDist`], [`hsMap`], [`barrier`]*) as data frames. If you remember the previous sections of this user guide (and sure enough you do, don't you?), you will know that for these parameters, MigClim supports both data frame and raster formats (what actually happens is that if you give the input in form of a data frame, MigClim will internally convert it into an ascii grid format). So here is the same call to the* `MigClim.migrate()` *function but using raster format as input. Remember that you have to give the name of the input raster files, not the actual raster. Also note that the code below assumes that you have converted the data to ascii grids, as shown in the previous section of this tutorial.*

```
N <- MigClim.migrate(iniDist="InitialDist", hsMap="hsmap", rcThreshold=500,
     envChgSteps=5, dispSteps=5, dispKernel=c(1.0,0.4,0.16,0.06,0.03),
     barrier="Barrier", barrierType="strong",
     iniMatAge=1, propaguleProd=c(0.01,0.08,0.5,0.92),
     lddFreq=0.1, lddMinDist=6, lddMaxDist=15,
     simulName="MigClimTest", replicateNb=3, overWrite=TRUE,
     testMode=FALSE, fullOutput=FALSE, keepTempFiles=FALSE)
```

Interpreting MigClim's output

Hopefully your simulation completed successfully, and you should see the following message displayed in your R console:

```
Simulation MigClimTest completed successfully. Outputs stored in [...]/MigClimTest
```

where [...] contains the path where your output directory is located. Looking inside the "MigClimTest" directory, you should find the following 12 files:

MigClimTest_params.txt : The "_params.txt" file contains a list of all the parameter values that were passed to the `MigClim.migrate()` function. You can always check this to remember which parameter values were used for a given simulation.

MigClimTest1_stats.txt, MigClimTest2_stats.txt, MigClimTest3_stats.txt, MigClimTest_stats.txt : The "_stats.txt" files contain the detail of how many cells were colonized or lost during each dispersal step. They also indicates the numbers of cells that would be occupied under the assumption of unlimited or no-dispersal for each dispersal step, as well as how many events of long distance dispersal were successful at each dispersal step. More detailed explanations on the content of the "_stats.txt" files are given later in this document.
Note that the "MigClimTest1_stats.txt", "MigClimTest2_stats.txt" and "MigClimTest3_stats.txt" files contain these information for respectively the 3 replicates of the simulation. The "MigClimTest_stats.txt" contains the average values obtained from the replicates.

MigClimTest1_summary.txt, MigClimTest2_summary.txt, MigClimTest3_summary.txt, MigClimTest_summary.txt : The "_summary.txt" files contain the same information than the "_stats.txt" files but, instead of having this information detailed for each dispersal step, these files only provide the totals over the entire simulation. The "MigClimTest1_summary.txt", "MigClimTest2_summary.txt" and "MigClimTest3_summary.txt" contain the values for respectively the 3 replicates. "MigClimTest_summary.txt" contains the average of all replicates.

MigClimTest1_raster.asc, MigClimTest2_raster.asc, MigClimTest3_raster.asc : These files are ASCII grid raster that contain the final state of the simulation. The values of the cell are coded so that it is possible to know which cell was colonized (or lost) during which dispersal step. Detailed explanations on how to read the values are given later in this document. You can display any of these maps using the `MigClim.plot()` function.

Note: if you perform a simulation without replication then you will have only one "_stats.txt," one "_summary.txt" and one "_raster.asc" file as output. If you perform a simulation with a larger number of replicates than 3, then you will obtain more "_stats.txt", "_summary.txt" and "_raster.asc" files.

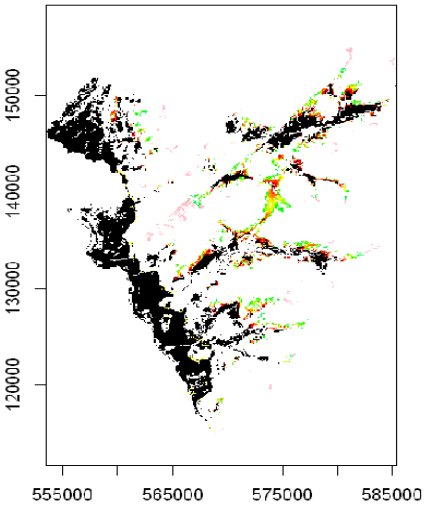
Understanding the MigClim output " raster.asc" files

As mentioned just above, the `MigClim.migrate()` function will output one (or several, if you chose more than 1 replication) ASCII grid files. These files have a name ending in "_raster.asc". You can visualize them in R, but also in a variety of other GIS software (e.g. ArcGIS from ESRI).

To display a MigClim output raster in R, you can use the `MigClim.plot()` function. Here is an example of how to display the "MigClimTest1_raster.asc" file that we obtained from our tutorial simulation:

```
MigClim.plot(asciiFile="MigClimTest/MigClimTest1_raster.asc",
             outDir="", fileFormat="inR", fullOutput=FALSE)
```

Copy/paste the above line in your R console and you should see the following image appear:

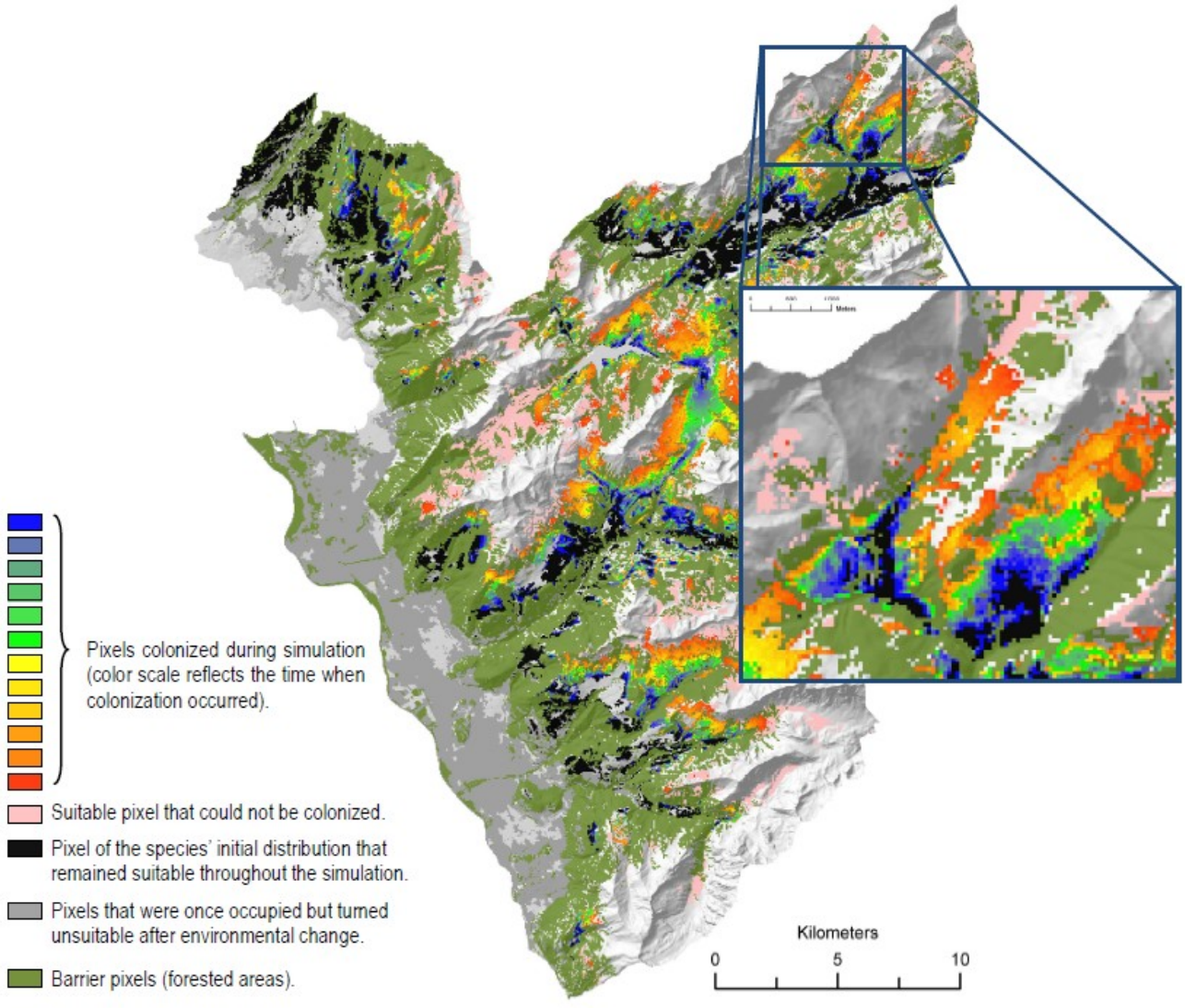


The map displays the projected distribution of our test species at the end of the simulation. The table below explains the interpretation of the different cell values and their color coding. The colors indicated in brackets in the first column of the table refer to the color of the cells when displaying a map using the `MigClim.plot()` function.

Cell value	Signification
0 [no color]	Cells that have never been occupied and are unsuitable habitat at the end of the simulation. There are several reasons why a cell can remain non-colonized: - The cell has remained unsuitable during the entire simulation. - The cell is part of a barrier. - The cell colonization has failed for dispersal-limitation related reasons.
1 (black)	Cells that belong to the species' initial distribution and that have remained occupied during the entire simulation.
1 < value < 30'000 [colored cells except for pink]	Positive values greater than 1 but smaller then 30'000 represent cells that have been colonized during the simulation and that remain occupied at the end of the simulation. The value of the cell allows to determine the dispersal step during which it was colonized using the following code: each environmental change step is given a value of 100 and each dispersal step a value of 1. Here are some examples: 101 = 1st dispersal step of 1st environmental change step (1 × 1 + 1 × 100 = 101). 102 = 2nd dispersal step of 1st environmental change step (2 × 1 + 1 × 100 = 102). 504 = 4th dispersal step of 5th environmental change step (4 × 1 + 5 × 100 = 504). 1003 = 3rd dispersal step of 10th environmental change step (3 × 1 + 10 × 100 = 101).
30'000 [pink]	Cells that are potentially suitable (i.e. habitat is favorable) but that were not colonized due to dispersal limitations. These cells represent the difference between the unlimited dispersal scenario and the current simulation.
Value < 1 [grey]	Negative values indicate cells that were once occupied but have become decolonized, because their habitat has turned unsuitable. Their absolute value allows determining the exact time when they have been decolonized using the same code as explained just above.

Note: If you are running a simulation without environmental change, then there will be no negative values in the output.

Note: parameters of the `MigClim.plot()` *function:*
In the example above, the output ascii grid file was displayed in the R console because we set the parameter [`fileFormat="inR"`]. *If you prefer to save the image directly as a "jpeg" or "png" file, you can do that by specifying* [`fileFormat="jpeg"`] *or* [`fileFormat="png"`]. *The file will be saved in the same directory as the "_raster.asc" file to which it corresponds. If you prefer to have the file saved in another directory then you can specify that in the* [`outDir`] *parameter (set* `outDir="any output directory"`*).*
The [`asciiFile`] *parameter is where you enter the name of the "_raster.asc" ascii grid that you wish to display. If the file is not located in your current working directory, then you should give the relative path to that directory, e.g.* `asciiFile="MigClimTest/MigClimTest1_raster.asc"`. *Finally, if you set* [`fullOutput=TRUE`], *then you will get maps not only for the final state of your simulation, but also for every intermediate dispersal step of the simulation. Note that this later option only works if you have run your simulation with* [`fullOutput=TRUE`] *in the* `MigClim.migrate()` *function (otherwise the required files will not have been produced).*
The color coding of the pixels given by the `MigClim.plot()` *function is explained in the table above.*



Interpretation of MigClim's text file outputs

Now that we have looked at all the raster outputs we will delve into the text files outputs. The text files contain more information than the raster files, but that information is no longer spatially explicit.

"_stats.txt" output files

The "_stats.txt" files contain the complete output of your MigClim simulation. They are simple tab delimited text files and can thus be opened with any basic text editor, or with a spreadsheet program such as OpenOffice/LibreOffice "Calc" or Microsoft "Excel". The files can also be read as a table in R:

```
statFile <- read.table("MigClimTest/MigClimTest_stats.txt", h=TRUE)
statFile
```

envChgStep	dispStep	stepID	univDispersal	noDispersal	occupied	absent	stepColonized	stepDecolonized	stepLDDsuccess
0	0	1	12914	12914	12914.00	117326.0	0.00	0	0.00
1	1	101	13933	12896	13605.33	116634.7	709.33	18	3.33
1	2	102	13933	12896	13651.67	116588.3	46.33	0	0.67
1	3	103	13933	12896	13681.33	116558.7	29.67	0	0.67
1	4	104	13933	12896	13721.00	116519.0	39.67	0	0.67
1	5	105	13933	12896	13751.00	116489.0	30.00	0	1.33
2	1	201	14576	12883	14151.00	116089.0	413.00	13	1.33
2	2	202	14576	12883	14200.67	116039.3	49.67	0	1.33
2	3	203	14576	12883	14236.33	116003.7	35.67	0	0.67
2	4	204	14576	12883	14277.67	115962.3	41.33	0	1.00
2	5	205	14576	12883	14308.00	115932.0	30.33	0	1.00
3	1	301	15066	12868	14653.00	115587.0	360.00	15	2.00
3	2	302	15066	12868	14707.67	115532.3	54.67	0	1.33
3	3	303	15066	12868	14741.67	115498.3	34.00	0	1.00
3	4	304	15066	12868	14769.33	115470.7	27.67	0	0.33
3	5	305	15066	12868	14784.67	115455.3	15.33	0	1.00
4	1	401	15722	12838	15208.67	115031.3	454.00	30	3.00
4	2	402	15722	12838	15255.33	114984.7	46.67	0	2.00
4	3	403	15722	12838	15283.00	114957.0	27.67	0	1.00
4	4	404	15722	12838	15310.00	114930.0	27.00	0	0.33
4	5	405	15722	12838	15336.00	114904.0	26.00	0	1.00
5	1	501	16340	12775	15747.00	114493.0	474.00	63	1.67
5	2	502	16340	12775	15785.33	114454.7	38.33	0	1.33
5	3	503	16340	12775	15818.00	114422.0	32.67	0	2.00
5	4	504	16340	12775	15844.33	114395.7	26.33	0	1.00
5	5	505	16340	12775	15866.00	114374.0	21.67	0	0.00

The first line of the table represents the initial state of the simulation. Each subsequent line represent the state of the simulation at the end of each dispersal step (the table has thus 26 rows: the initial state + 25 dispersal steps). The last row of the table (in this case after 25 dispersal steps) represent the final state of the simulation.

The following table will give you a detailed description of what the values of each column mean:

Column Name	Description
envChgStep	Number of the environmental change step. In the case of our simulation, we have implemented environmental change every 5 dispersal steps, therefore you can see that there are always 5 successive lines with the same "envChgStep" value. *Note: The fist line has a "envChgStep" value of "0" because this line contains the information regarding the initial distribution of the species, before the simulation was started.*
dispStep	Number of the dispersal step. In the case of our simulation, we have implemented 5 dispersal steps within each environmental change step, and therefore you can see that this column always repeats the numbers 1 to 5 every 5 lines. *Note: The fist line has a "dispStep" value of "0" because this line contains the information regarding the initial distribution of the species, before the simulation was started.*
stepID	This column indicates the "coded" value associated to each dispersal step. The "stepID" value of each line is computed as follows: stepID = envChgStep number * 100 + dispStep number (e.g. line 2 has a value of 101, that is computed as follows: 101 = 1 * 100 + 1). The "stepID" values correspond to the pixel values in the output ascii grid files (they use the same coding system).

The first line of this column contains the number "1" indicating that this line contains the information about the initial distribution of the species, before the simulation was started.

univDispersal This column contains the number of cells that would be occupied in the case of the "Unlimited Dispersal" scenario (i.e. this assumes that the species has unlimited dispersal ability and colonizes any cells that is suitable, regardless of its location) at the end of each dispersal step. The first line of this column contains the number of cells of the species' initial distribution.

Because we have implemented a change in the habitat suitability map every 5 years, you will notice that values in this column only changes every 5th line. This makes sense, because under the unlimited dispersal scenario, all cells that are suitable are colonized immediately when they become suitable. Thus, all suitable cells become colonized during the first dispersal step when the habitat becomes suitable and there is nothing left to colonize during the remaining four dispersal steps of the environmental change step.

noDispersal This column contains the number of cells that would be occupied in the case of the "No Dispersal" scenario (i.e. a scenario in which we consider the dispersal of a species as null) at the end of each dispersal step. The first line of this column contains the number of cells of the species' initial distribution.

Because we have implemented change in the habitat suitability every 5 years, you will notice that values in this column only change every 5th line. This makes sense, because under the no dispersal scenario, all cells are lost immediately when they turn unsuitable. Thus, all cells that will be lost due to environmental change are lost during the first dispersal step when the habitat becomes unsuitable and nothing more can be lost during the remaining four dispersal steps.

Note: by definition, under the "No Dispersal" scenario, a species can never increase in distribution. Thus the values in this column will only decrease, and never increase.

occupied Number of cells that are in an "occupied" state at the end of the given dispersal step.

absent Number of cells that are in an "unoccupied" state at the end of the given dispersal step.

stepColonized Number of cells that turned into "occupied" state (= number of cells that got colonized) during the given dispersal step.

stepDecolonized Number of cells that turned into "unoccupied" state during the given dispersal step.

stepLDDsuccess Number of successful LDD events (i.e. LDD events that led to the colonization of a cell) that occurred during the given dispersal step.

"_summary.txt" output files

As their name suggests, "_summary.txt" files contain a summary of each simulation's output. Overall they give similar information as found in the "_stats.txt" files, except that they do not give the details for each dispersal step but only for the final state of the simulation.

Again, the "_summary.txt" files are tab delimited text files and can be read with any text or spreadsheet editor. You can also load them into R as tables:

```
summaryFile <- read.table("MigClimTest/MigClimTest_summary.txt", h=TRUE)
summaryFile
    simulName iniCount noDispCount univDispCount occupiedCount absentCount totColonized totDecolonized
 MigClimTest1    12914       12775         16340         15881      114359         3106            139
 MigClimTest2    12914       12775         16340         15877      114363         3102            139
 MigClimTest3    12914       12775         16340         15840      114400         3065            139
  MigClimTest    12914       12775         16340         15866      114374         3091            139

totLDDsuccess runTime
           26    3.00
           35    2.00
           32    3.00
           31    2.67
```

Notice that this file has 4 rows: the first 3 rows contain the values for each of our simulation replicates (we choose to do 3 replicates), while the last row gives the average of the 3 replicates.

If you chose to do a simulation without replication [`replicateNb=1`] then the "_summary.txt" file will have only 1 row. If you choose to do more than one replicate, then the"_summary.txt" output file will contain [`replicateNb`]+1 rows (the results for each replicate plus the average).

The following table gives a detailed description of each column of a "_summary.txt" file:

Column Name	Description
simulName	Name of the simulation.
iniCount	Number of cells of the species' initial distribution.
noDispCount	Number of cells colonized at the end of the simulation under the "No Dispersal" scenario (i.e. this assumes that the species cannot disperse and hence cannot colonize any new habitat).
univDispCount	Number of cells colonized at the end of the simulation under the "Unlimited Dispersal" scenario (i.e. this assumes that the species has unlimited dispersal ability and colonizes any cells that is suitable, regardless of its location).
occupiedCount	Number of cells in an "occupied" state at the end of the simulation (i.e. the potential distribution of your species given the implemented dispersal restrictions).
absentCount	Number of cells in "unoccupied" state at the end of the simulation.
totColonized	Total number of cells colonized during the entire simulation.
totDecolonized	Total number of cells lost due to habitat turning unfavorable during the entire simulation.
totLDDsuccess	Total number of successful LDD events (i.e. LDD events that led to the colonization of a cell) that occurred during the entire simulation.
runTime	Total run time of simulation in seconds (rounded to the nearest second, so if the simulation takes less than 1 second you might see a value of 0).

"_params.txt" files

The "_params.txt" file simply contains a log of all the parameter inputs that were used for a given simulation, i.e. all the inputs values of the `MigClim.migrate()` function. These files can be helpful in case you forgot which parameters values were used in a simulation. Each row of the file contains one parameter.