# 'SiMRiv'
# (version 1.0.0)
# An R package for simulation and analysis of spatially-explicit individual multi-state (animal) movements in any landscape

Miguel Porto, Lorenzo Quaglietta

July 27, 2017

## Contents



## 1 Introduction

The study of animal movement behaviour, recently reframed in the 'Movement Ecology' paradigm (Nathan 2008), is key to a variety of fields including landscape ecology, species distribution modeling, and population size estimation among others (Turchin 1998; Nathan 2008; Royle et al 2014). A common feature in the modeling of animal movement data, used also in fields as varied as anthropology (Raichlen et al 2014), paleontology (Sims et al 2014), cell biology (Dickinson & Tranquillo 1993), tumour angiogenesis (Plank & Sleeman 2003), and molecular ecology (Hoban 2014), is the comparison between real and simulated, spatially-explicit, individual trajectories (Sims et al 2008; Humphries et al 2010; Viswanathan et al 2010; reviews in Patterson et al 2008, Schick et al 2008, Smouse et al 2010, and Hooten et al 2017). Despite the increasing availability of (GPS) telemetry data and of powerful statistical and modeling approaches for the study

of movement, lack of suitable analytical software and computational power still constrains our comprehension of animal movement patterns (Morales et al 2010; Calabrese et al 2016; Michelot et al 2016; Hooten et al 2017).

In particular, we note that while software for simulating individual trajectories of species that move without restrictions in the landscape (species moving in a bi-dimensional homogeneous space) is available (e.g., Calenge et al 2009; Beyer 2012012, used e.g. in Fortin et al 2005; Humphries et al. 2013; Michelot et al 2016), we are aware of no appropriate software for simulating spatially-explicit, individual movement trajectories of species inhabiting (or intensively using) linear or dendritic landscapes (e.g., trouts or otters in rivers, humans along roads) or somehow conditioned by landscape heterogeneity (e.g., species moving along corridors or avoiding certain habitat features). This lack of software may at least partly explain why, while a number of studies focused on marine environments (e.g., Sims et al 2008; Humphries et al 2010; Viswanathan 2010; Reynolds 2014; Hussey et al 2015), movement ecology has been poorly investigated in freshwater linear landscapes (e.g., river networks), also called dendritic (Peterson et al 2013) - or highly structured (Sutherland et al 2014) - ecological networks (DENs). Knowledge on the movement ecology of freshwater aquatic and semiaquatic species is in fact particularly scant.

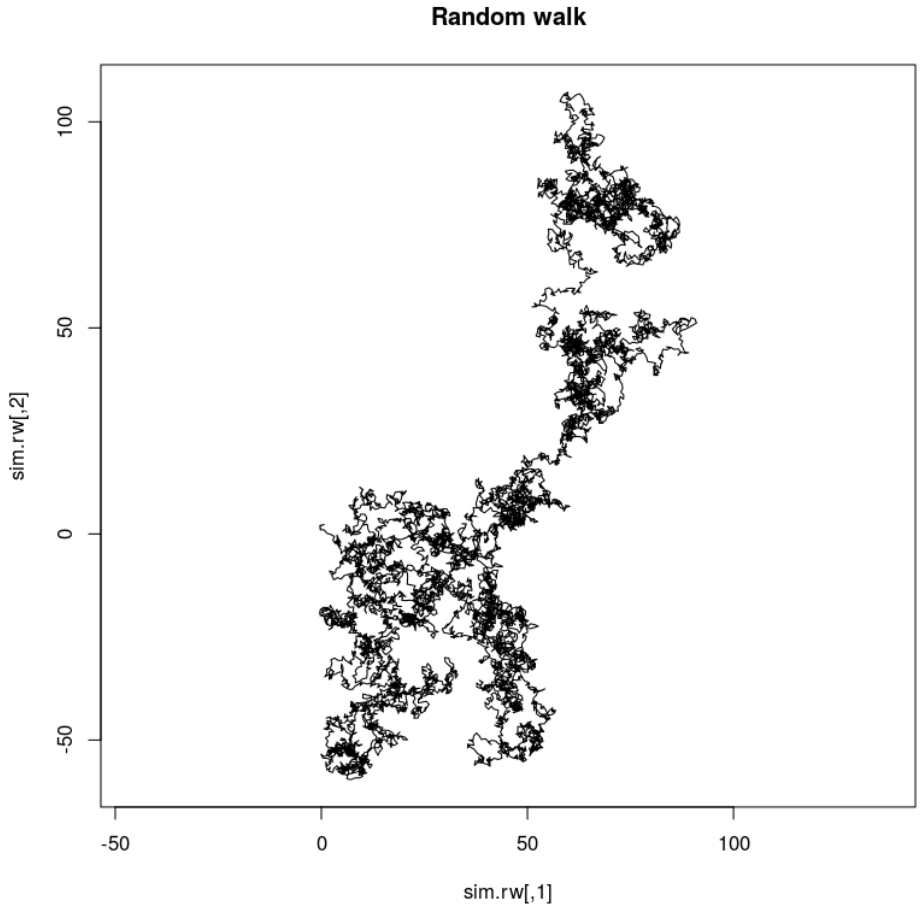
SiMRiv was specifically conceived to allow simulating individual-based, spatially-explicit movements in river networks and heterogeneous landscapes, thus filling the above-highlighted gap of software simulating movements in DENs. More importantly, the software was thought and developed to be highly flexible, allowing to simulate trajectories of any species living in any landscape, irrespective of its complexity (including totally homogeneous landscapes), with precisely the same algorithm. It thus allows to develop realistic movement models using multi-state markov models, while taking into account environmental complexity.

In this document, we: (i) introduce how to prepare the data for the simulations; (ii) report the main functionalities of the software; (iii) introduce some examples of use of the software; and (iv) provide an overview on main software's potential applications, expected outcomes, and features we would like to improve or develop. We assume that you are somehow familiar with the basic concepts of Movement Ecology (see reviews on this, as well as on the use of random walk models in Ecology, in Nathan 2008, Codling et al 2008, Patterson et al 2008, Schick et al 2008, Smouse et al 2010), as well as with the R language and environment. In case you need help for the 'raster' package, you may want to refer to its vignette. This is a work in progress, **report of any bug or suggestion is warmly welcomed.**

## 2 Conducting basic simulations

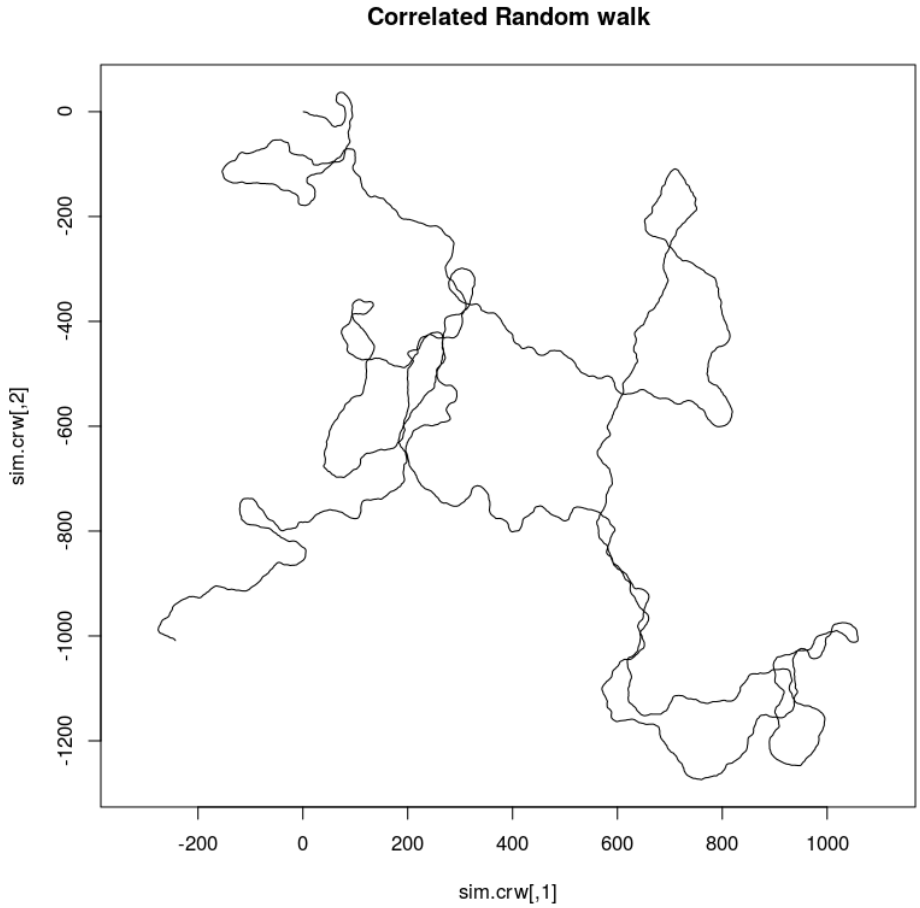
In the simplest form, conducting simulations is straightforward. Using the default parameters setting, individuals will be simulated in an homogeneous environment, will start at coordinates (0, 0), and have a unit step length for all states. You only have to provide the movement type of each state. This is done by defining a `species`. Let's first look at the simplest case, a simple Random Walk (cf. Turchin 1998):

```
> # define a species with a single-state movement type
> # characterized by a random walk
> rand.walker <- species(state.RW())
> # simulate one individual of this species, 10000 simulation steps
> sim.rw <- simulate(rand.walker, 10000)
> # plot trajectory
> plot(sim.rw, type = "l", asp = 1, main = "Random walk")
```




`state.RW` is a shortcut function for defining a totally random movement state, with unit step length. Now let's try with a more interesting movement type, a simple Correlated Random Walk, where the direction taken in one step is correlated with the direction of the previous step. For this you have the shortcut function `state.CRW`, to which you must provide the amount of correlation, more precisely, the turning angle concentration parameter of the wrapped normal distribution (below we use 0.98). This parameter varies between [0, 1], being 0 no correlation (yielding a random walk state) and 1, full correlation (step direction does not change, it's always the same as the previous step). Obviously, `state.RW()` is a shortcut for `state.CRW(0)`.

```
> # define a species with a single-state movement type characterized
> # by a correlated random walk with concentration=0.98
> c.rand.walker <- species(state.CRW(0.98))
> # simulate one individual of this species
> # 10000 simulation steps
> sim.crw <- simulate(c.rand.walker, 10000)
> plot(sim.crw, type = "l", asp = 1, main = "Correlated Random walk")
```




Additionally, you have a third shortcut `state.Resting()`, which corresponds to a state where the individual is stopped. These three shortcut functions are the bricks with which you construct any complex multistate movement. You can combine them with `+` to define a multistate movement, see next topic.

All simulations above were conducted with the step length = 1, which is the default. You may set the desired step length for any state simply by adding a number to the state definition, or you may set the same step length for all states by adding a number to the species itself.

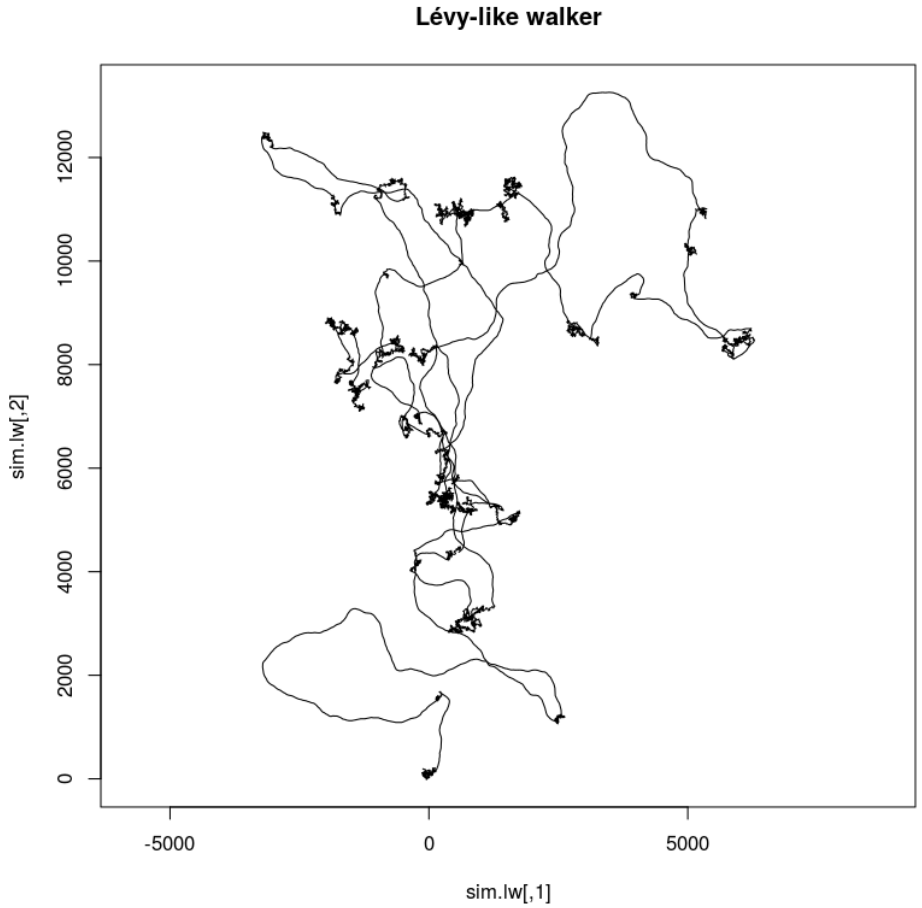
```
> # define a species with a correlated random walk
> # and step length = 15
> c.rand.walker.15 <- species(state.CRW(0.98) + 15)
> # which, in single-state species, is the same as:
> c.rand.walker.15 <- species(state.CRW(0.98)) + 15
```

# 3 Multi-state movement simulations

Multi-state movements require specifying the probabilities of transition between the movement states. This transition matrix, which is a parameter you specify when defining the species, has an overwhelming importance in the final movement pattern. Formally, it is just a square matrix with all values in the range [0, 1], and whose rows must sum to one (but not columns). It should be read row-wise, that is, the cell at row 2 column 3 defines the probability of the individual changing from state #2 to state #3.

As an example, let's simulate a two-state movement, a Lévy-like walker, i.e. a walker who alternates between random walks and correlated random walks. We can define the multi-state movement by combining the two states like `state.RW() + state.CRW(0.98)`. Then, we have to provide the probabilities of changing from state #1 to state #2 and from state #2 to state #1. A convenience function, `transitionMatrix`, is provided to build such matrix. In the example below, we define the probabilities: #1 -> #2: 0.005; #2 -> #1: 0.01 We also set the step length for both states to `25`.

```
> # a Lévy walker can be approximated by a two-state walker
> # composed of a random walk state and a correlated
> # random walk state.
> levy.walker <- species(state.RW() + state.CRW(0.98)
+   , trans = transitionMatrix(0.005, 0.01)) + 25
> sim.lw <- simulate(levy.walker, 10000)
> plot(sim.lw, type = "l", asp = 1, main = "Lévy-like walker")
```




The same logic applies to any number of states, but obviously, the transition matrix quickly gets large. See `help(transitionMatrix)` for details.

# 4 Parameterizing the simulation

In the current version, we provide an experimental method for numerically approximating simulation input parameters from real trajectory data, using an heuristic optimization. This is

done with the function `adjustModel`. This function numerically finds combinations of input parameters that minimize the differences between simulated trajectories and the real trajectory. See the working example in `help(adjustModel)`.

# 5 Simulating in heterogeneous environments

`SiMRiv` can be used to simulate movements in an homogeneous environment, in which case you don't need to import data. However, if you want to conduct simulations in heterogeneous environments (which includes riverscapes), a resistance raster must be given as input to the simulation procedure. This raster defines, for all pixels in space, the propensity of the individuals to move within/into each pixel (see details and discussion in `help(simulate)`). Any object of class `RasterLayer` (from the `raster` package) with values in the range [0, 1] can be used, where 0 means a pixel with no resistance and 1 a pixel with 'infinite' resistance (i.e. where the individual cannot move).

## 5.1 Data import

`SiMRiv` provides a helper function `resistanceFromShape` to aid the conversion from vector data (line and polygon shapefiles) to a resistance raster, optionally combining multiple rasters into one (e.g., a physical resistance raster and a resource distribution raster). It is basically a wrapper for the functionality provided by the `raster` package, combining multiple features in one function:

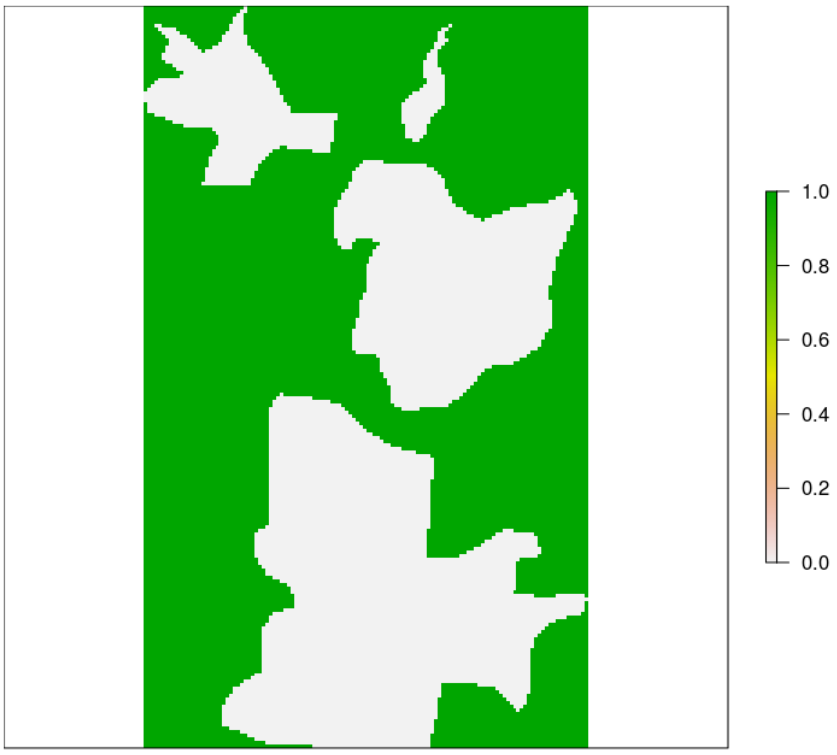
1. Rasterize polygon or line shapefiles with a user-given pixel resolution
2. Assign resistance values to a categorical field in the shapefile or directly read values from a numerical field
3. Assign a background value for areas not covered by the polygons or lines
4. Create a buffer around lines (or polygons), optionally variable in size
5. Stack multiple shapefiles into one combined raster

Its use is exemplified below.

### 5.1.1 Importing shapefiles to use as resistance raster

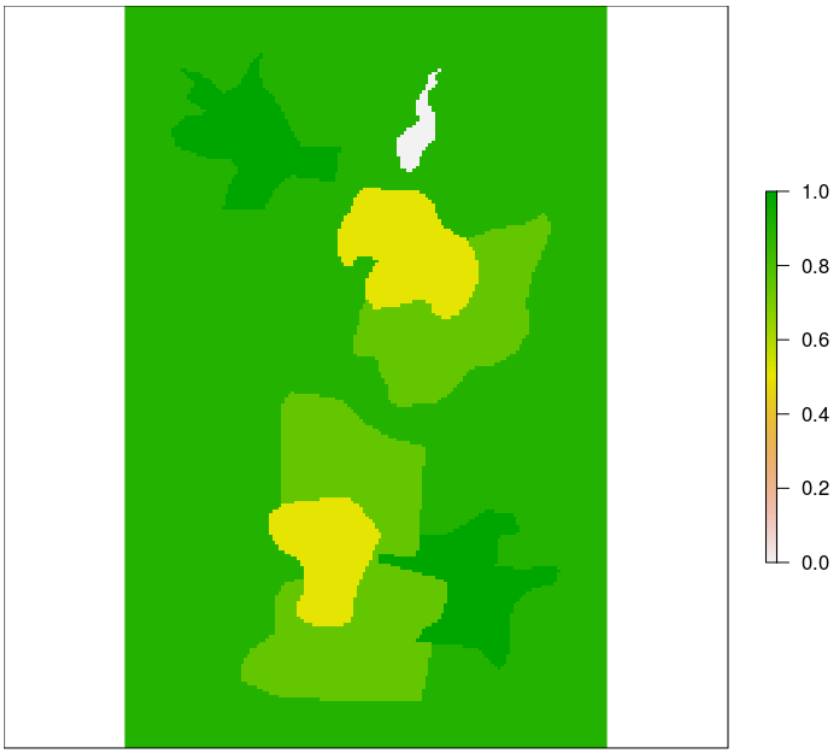
The following example is the simplest case, it creates a binary resistance raster with 100m resolution, assigning resistance 0 to areas covered by polygons in the given shapefile and resistance 1 to areas not covered. You can change these values with the parameters `field` (defaults to 0) and `background` (defaults to 1). The same works for line shapefiles.

```
> resistance <- resistanceFromShape(
+   system.file("doc/landcover.shp", package="SiMRiv")
+   , res = 100)
> plot(resistance, axes = F)
```



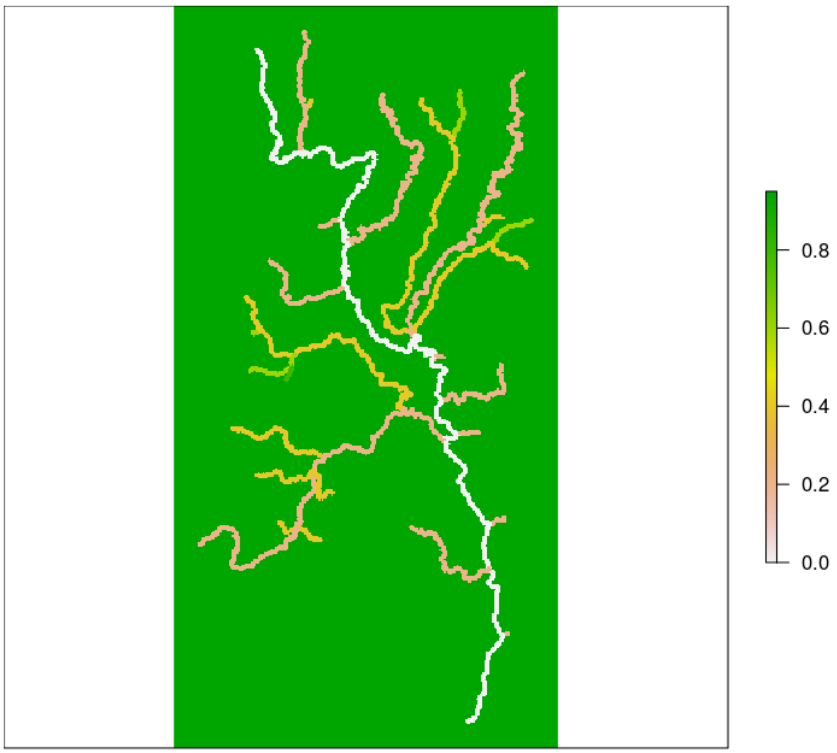

The following example creates a resistance raster with 100m resolution by importing a polygon shapefile while assigning different resistance values to each land cover class (given by the `mapvalues` parameter) provided in the shapefile field `coverclass`. All pixels not covered by a polygon are assigned a resistance of 0.9. Note that you can add a margin around the extent of the shapes (here, we used a 3000 m margin), to minimize the boundary effect in simulations: during simulations, all area outside the extent of the resistance raster is given resistance 1, which makes the simulated individuals never move out of this rectangle, thus creating a boundary effect.

```
> resistance <- resistanceFromShape(
+   system.file("doc/landcover.shp", package="SiMRiv")
+   , res = 100, field = "coverclass", mapvalues = c(
+     "forest" = 0.5, "urban" = 1, "dam" = 0
+     , "shrubland" = 0.75)
+   , background = 0.9, margin = 3000)
> plot(resistance, axes = F)
```



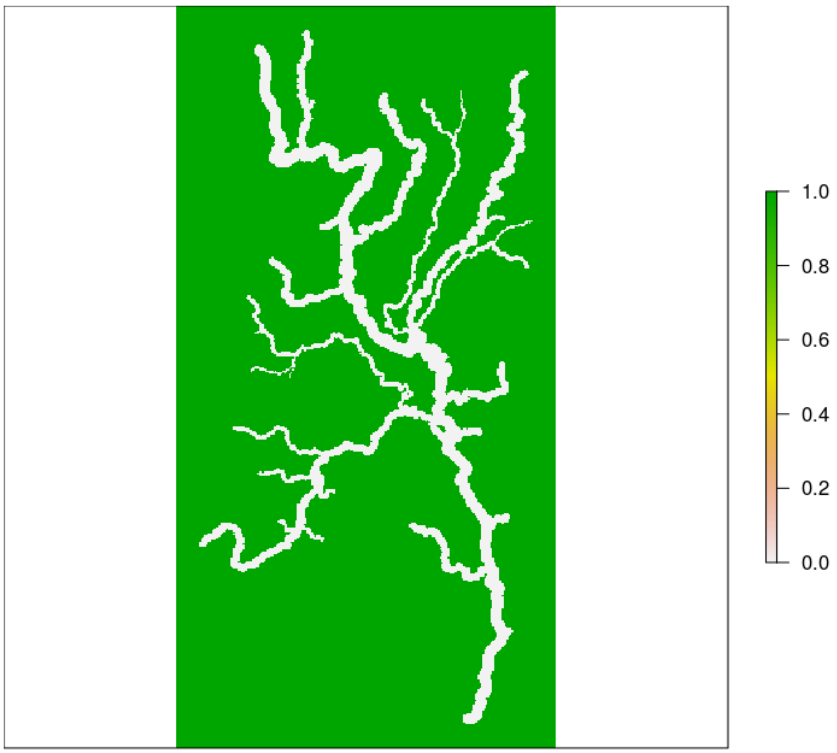

The same applies for line shapefiles. In this case, it is useful to make a buffer around lines, otherwise the rasterized version may be too thin for the simulation to adequately proceed. Note that, by default, if no buffer is used, the rasterization of lines 'projects' the lines into all cells that are touched by the line (see `help(rasterize)` of the `raster` package), which results in 1-pixel wide features. This should be avoided; always make sure that the thinnest line elements are at least 3 pixel wide in the resistance raster. In this example with a river (using a buffer of 150m), resistance is manually assigned according to the river order field '`Order`' of the shapefile (higher resistance values for lower stream order sectors - i.e., the species is assumed to use main river sectors more than tributaries):

```
> resistance <- resistanceFromShape(
+   system.file("doc/river-sample.shp", package="SiMRiv")
+   , res = 100, field = "Order", mapvalues = c("2" = 0
+     , "3" = 0.2, "4" = 0.4, "5" = 0.6, "6" = 0.8)
+   , buffer = 150, background = 0.95, margin = 3000)
> plot(resistance, axes = F)
```




A better, and possibly more realistic, option for rivers, instead of varying the resistance, would be to use a buffer proportional to the river order (alternatively, both solutions may be applied). In that case, we're better off loading the shape separately:

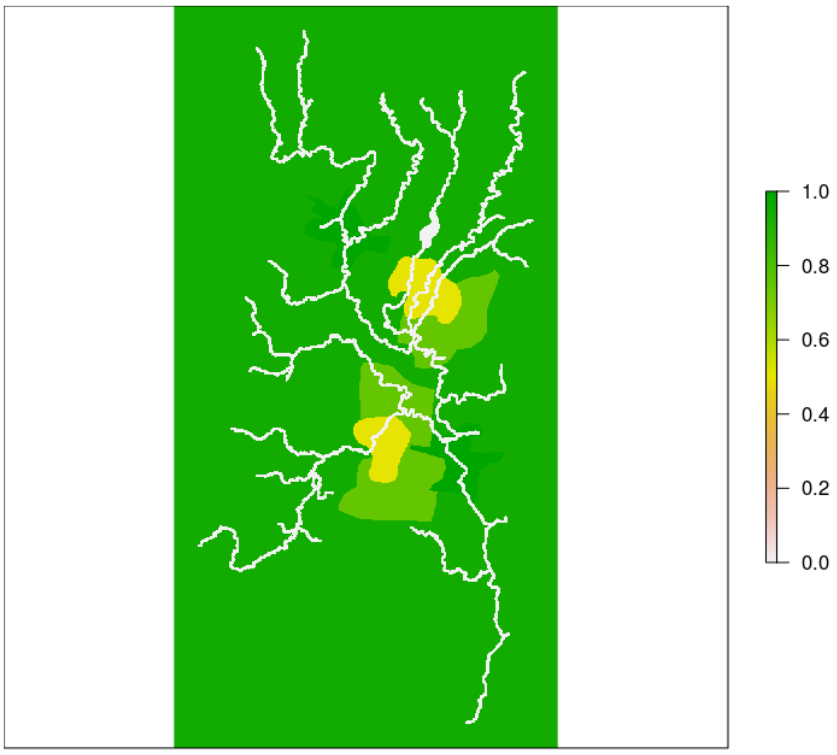
```
> # load shapefile
> river.shape <- shapefile(system.file("doc/river-sample.shp", package="SiMRiv"))
> # below you can provide the shapefile filename, or the
> # R shapefile object itself
> resistance <- resistanceFromShape(river.shape, res = 100
+   , buffer = (9 - river.shape@data$Order) ^ 3
+   , background = 0.95, margin = 3000)
> # buffer here is just some magical function to convert river
> # order into a meaningful value in the [0, 1] range!
>
> plot(resistance, axes = F)
```




### 5.1.2 Combining multiple shapefiles or rasters in one resistance raster

Often one needs to combine data from different shapefiles into a combined resistance raster. The following example places the river on top of the land cover shape, using the parameter `baseRaster`. The extent is automatically adjusted to fit all provided shapefiles' extents (plus the optional margin). Note also that, if we want a constant resistance value for all lines/polygons in a shape, we may specify the value with the parameter `field` (as is done below in `river.landcover`), instead of giving a field name (as is done below in `landcover`).

```
> landcover <- resistanceFromShape(
+   system.file("doc/landcover.shp", package="SiMRiv")
+   , res = 50, field = "coverclass", mapvalues = c(
+     "forest" = 0.5, "urban" = 1, "dam" = 0
+     , "shrubland" = 0.75), background = 0.95)
> river.landcover <- resistanceFromShape(
+   system.file("doc/river-sample.shp", package="SiMRiv")
+   , baseRaster = landcover, buffer = 100, field = 0
+   , background = 0.95, margin = 3000)
> plot(river.landcover, axes = F)
```




## 5.2 Conducting simulations on a resistance raster

When simulating in heterogeneous environments, one further parameter must be taken into account, which is how the species perceives the environment to make its movement decisions. This is implemented as the concept of **perceptual range**: the area around the individual's current position, which it takes into account for making decisions (Lima & Zollner 1996). As of version 1.0.0, this is implemented as a circle centered on the individual's position, with a user-specified radius.

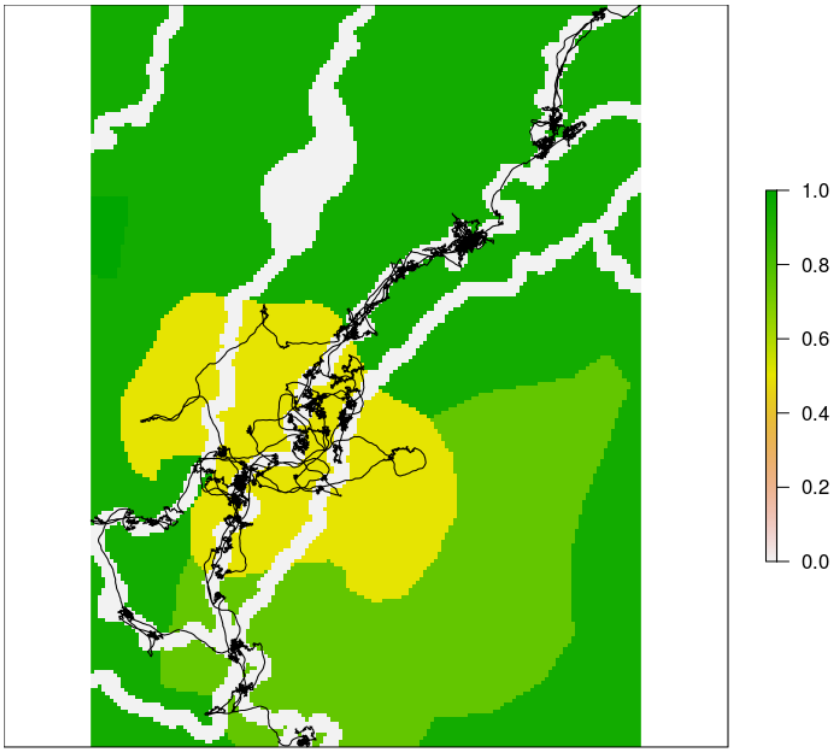
During simulations, at each step, the turning angles of the individuals will be biased towards lower resistance areas, and this is evaluated only within the radius of the perceptual range. This parameter also has an overwhelming importance in the resulting movement patterns. A species with a wider perceptual range will be attracted to lower resistance areas, even if it has to cross high resistance areas, because it 'sees' farther. Conversely, a species with a small perceptual range will not 'see' those lower resistance areas, so it will be detained by high resistance areas. This can have important consequences, for example, when simulating amphibious species in a river: the size of the perceptual range will determine how likely the individuals will take overland shortcuts between river segments.

In order to allow more flexibility, the perceptual range is an attribute of each state (not of the species). For ease of use, the size of the perceptual range of a state can be set simply by multiplying the state by a number, for example `state.RW() * 1000` will define a random walk state with a perceptual range of 1000 m. If you want to set the same value for all the states of a species, you can directly multiply the species by a number, e.g. `levy.walker * 1500` will set

the perceptual range of both states of the Lévy-like walker to 1500 m.

In the next example, we simulate a Lévy-like walker in a riverscape, assuming an amphibious species, i.e. that moves mainly in water but may also walk overland if needed. This example also demonstrates how to set the starting coordinates to a random pixel with given resistance value(s).

```
> # set starting coordinates anywhere within the river
> init = xyFromCell(river.landcover, sample(which(values(river.landcover) == 0), 1))
> # adding a number to a species is a shortcut for setting
> # the step lengths of all states
> # multiplying is a shortcut for setting the perceptual range radius
> levy.walker <- (levy.walker + 15) * 1000
> sim.lw.river <- simulate(levy.walker, 40000
+   , resist = river.landcover, coords = init)
> # plot resistance
> plot(river.landcover, axes = F
+   , ylim = range(sim.lw.river[, 2]), xlim = range(sim.lw.river[, 1]))
> # plot trajectory on top of resistance
> lines(sim.lw.river)
>
```



# 6 Advices for conducting bias-free simulations

## 6.1 Step lengths and resolution

When using resistance rasters, it is important to adequately choose the step length. You should consider the size of the raster features when choosing an appropriate step length: if the step length is larger than the smaller features (say, the width of a river), the individuals will jump over those features, and the simulation will not be correct. Step length should always be small, close to the real steps of the simulated species, and much smaller than the smallest landscape features. This is because the `SiMRiv` simulation algorithm was developed in the logic of conducting infinitesimal simulations. If you want to have simulated data which is comparable to real tracking data, you must downsample the simulation results with the function `sampleMovement`.

# 7 References


Beyer, H.L. (2012). Geospatial modeling environment (version 0.7. 2.1). URL:< http://www. spatialecology. com/gme.

Calabrese, J. M., Fleming, C. H., & Gurarie, E. (2016). ctmm: an r package for analyzing animal relocation data as a continuous-time stochastic process. Methods in Ecology and Evolution, 7(9), 1124-1132.

Calenge, C., Dray, S., & Royer-Carenzi, M. (2009). The concept of animals' trajectories from a data analysis perspective. Ecological informatics 4, 34-41.

Codling, E. A., Plank, M. J., & Benhamou, S. (2008). Random walk models in biology. Journal of the Royal Society Interface, 5(25), 813-834.

Dickinson, R.B. & Tranquillo, R.T. (1993). Optimal estimation of cell movement indices from the statistical analysis of cell tracking data. AIChE Journal 39, 1995-2010.

Fortin, D., Beyer, H.L., Boyce, M.S., Smith, D.W., Duchesne, T., & Mao, J.S. (2005). Wolves influence elk movements: behavior shapes a trophic cascade in Yellowstone National Park. Ecology 86, 1320-1330.

Hoban, S. (2014). An overview of the utility of population simulation software in molecular ecology. Molecular ecology 23, 2383-2401.

Hooten, M. B., Johnson, D. S., McClintock, B. T., & Morales, J. M. (2017). Animal movement: statistical models for telemetry data. CRC Press.

Humphries, N.E., Queiroz, N., Dyer, J.R.M., et al. (2010). Environmental context explains Lévy and Brownian movement patterns of marine predators. Nature 465, 1066-1069. doi:10.1038/nature09116.

Humphries, N.E., Weimerskirch, H. & Sims, D.W. (2013). A new approach for objective identification of turns and steps in organism movement data relevant to random walk modelling. Methods in Ecology and Evolution 4, 930-938. DOI: 10.1111/2041210X.12096.

Hussey, N. E., Kessel, S. T., Aarestrup, K., Cooke, S. J., Cowley, P. D., Fisk, A. T., ... & Flemming, J. E. M. (2015). Aquatic animal telemetry: a panoramic window into the underwater world. Science, 348(6240), 1255642.

Lima, S. L., & Zollner, P. A. (1996). Towards a behavioral ecology of ecological landscapes. Trends in Ecology & Evolution, 11(3), 131-135.

Michelot, T., Langrock, R., & Patterson, T. A. (2016). moveHMM: An R package for the statistical modelling of animal movement data using hidden Markov models. Methods in Ecology and Evolution, 7(11), 1308-1315.

Morales, J.M., Moorcroft, P.R., Matthiopoulos, J. et al. (2010). Building the bridge between animal movement and population dynamics. Phil. Trans. R. Soc. B 365, 2289-2301.

Nathan, R. (2008). An emerging movement ecology paradigm. PNAS 105, 19050-19051.

Patterson, T.A., Thomas, L., Wilcox, C., Ovaskainen, O., & Matthiopoulos, J. (2008). State-space models of individual animal movement. Trends in ecology & evolution 23, 87-94.

Peterson, E.E., Ver Hoef, J.M., Isaak, D.J., et al. (2013). Modelling dendritic ecological networks in space: an integrated network perspective. Ecology Letters 16, 707-719.

Plank, M.J. & Sleeman, B.D. (2003). A reinforced random walk model of tumour angiogenesis and anti-angiogenic strategies. Mathematical Medicine and Biology 20, 135-181.

Raichlen, D.A., Wood, B.M., Gordon, A.D., Mabulla, A.Z., Marlowe, F.W. & Pontzer, H. (2014). Evidence of Lévy walk foraging patterns in human hunter-gatherers. Proceedings of the National Academy of Sciences 111, 728-733.

Reynolds, A. M. (2014). Mussels realize Weierstrassian Lévy walks as composite correlated random walks. Scientific reports, 4, 4409.

Royle, J.A., Chandler, R.B., Sollmann, R. & Gardner, B. (Eds) (2014). Spatial Capture Recapture. Academic Press.

Schick, R. S., Loarie, S. R., Colchero, F., Best, B. D., Boustany, A., Conde, D. A., ... & Clark, J. S. (2008). Understanding movement data and movement processes: current and emerging directions. Ecology letters, 11(12), 1338-1350.

Sims, D.W., Southall, E.J., Humphries, N.E., et al. (2008). Scaling laws of marine predator search behaviour. Nature 451, 1098-1102.

Sims, D.W., Reynolds, A.M., Humphries, N.E., Southall, E.J., Wearmouth, V.J., Metcalfe, B. & Twitchett, R.J. (2014). Hierarchical random walks in trace fossils and the origin of optimal search behavior. Proceedings of the National Academy of Sciences 111, 11073-11078.

Smouse, P. E., Focardi, S., Moorcroft, P. R., Kie, J. G., Forester, J. D., & Morales, J. M. (2010). Stochastic modelling of animal movement. Philosophical Transactions of the Royal Society of London B: Biological Sciences, 365(1550), 2201-2211.

Sutherland, C., Fuller, A. K., & Royle, J. A. (2015). Modelling non-Euclidean movement and landscape connectivity in highly structured ecological networks. Methods in Ecology and Evolution, 6(2), 169-177.

Turchin, P. (1998). Quantitative Analysis of Movement: measuring and modeling population redistribution in plants and animals. Sinauer Associates, Sunderland, MA.

Viswanathan, G.M. (2010). Ecology: Fish in Lévy flight foraging. Nature 465, 1018-1019.

# 8 To do list and opportunities for collaborations

The software in its current version does not allow for:

1. spatial bias (e.g. simulate a biased random walk);
2. separation between physical resistance and habitat suitability, which should have different effects on movement;
3. simulations of multiple interacting individuals;
4. improving current method for input parameter estimation (`adjustModel`) or assessing the applicability of other methods (cf. Michelot et al 2016)
5. other distributions for turning angles and steplength (currently, we use wrapped normal for the former and a fixed value for the latter);

**We highly encourage interested programmers to join our project on** GitHub and contribute by improving and further extending the package.