R as a Simulation Platform in Ecological Modelling

Thomas Petzoldt

Introduction

In recent years, the R system has evolved to a mature environment for statistical data analysis, the development of new statistical techniques, and, together with an increasing number of books, papers and online documents, an impressing basis for learning, teaching and understanding statistical techniques and data analysis procedures.

Moreover, due to its powerful matrix-oriented language, the clear interfaces and the overwhelming amount of existing basic and advanced libraries, R became a major platform for the development of new data analysis software (Tierney, 2003).

Using R firstly for post-processing, statistical analysis and visualization of simulation model results and secondly as a pre-processing tool for externally running models the question arose, whether R can serve as a general simulation platform to implement and run ecological models of different types, namely differential equation and individual-based models.

From the perspective of an empirical ecologist, the suitability of a simulation platform is often judged according to the following properties:

1. learning curve and model development time,
2. execution speed,
3. readability of the resulting code,
4. applicability to one special model type only or to a wide variety of simulation methods,
5. availability of libraries, which support model development, visualization and analysis of simulation results,
6. availability of interfaces to external code and external data,
7. portability and licensing issues.

While questions 5 to 7 can be easily answered in a positive sense for the R system, e.g. portability, free GNU Public License, the availability of external interfaces for C, Fortran and other languages and numerous methods to read and write external data, the remaining questions can be answered only with some practical experience. As a contribution, I present some illustrative examples on how ecological models can be implemented within R and how they perform. The documented code, which can be easily adapted to different situations will offer the reader the possibility to approach their own questions.

Examples

The examples were selected to show different types of ecological models and possible ways of implementation within R. Although realism and complexity are limited due to the intention to give the full source code here, they should be sufficient to demonstrate general principles and to serve as an onset for further work[1].

Differential equations

The implementation of first-order ordinary differential equation models (ODEs) is straightforward and can be done either with an integration algorithm written in pure R, for example the classical Runge-Kutta 4th order method, or using the `lsoda` algorithm (Hindmarsh, 1983; Petzold, 1983), which thanks to Woodrow Setzer are both available in the **odesolve**-library. Compared to other simulation packages this assortment is still relatively limited but should be sufficient for Lotka-Volterra type and other small and medium scale ecological simulations.

As an example I present the implementation of a recently published Lotka-Volterra-type model (Blasius et al., 1999; Blasius and Stone, 2000), the constant period – chaotic amplitude (UPCA) model. The model consists of three equations for resource u, herbivorous v, and carnivorous w animals:

$$\frac{du}{dt} = au - \alpha_1 f_1(u,v) \tag{1}$$

$$\frac{dv}{dt} = -bv + \alpha_1 f_1(u,v) - \alpha_2 f_2(v,w) \tag{2}$$

$$\frac{dw}{dt} = -c(w - w^*) + \alpha_2 f_2(v,w) \tag{3}$$

where f_1, f_2 represent either the Lotka-Volterra term $f_i(x,y) = xy$ or the Holling type II term $f_i(x,y) = xy/(1+k_i x)$ and w^* is an important stabilizing minimum predator level when the prey population is rare.

To run the model as an initial value simulation we must provide R with (1) the model written as an R-function, (2) a parameter vector, (3) initial (start) values, and (4) an integration algorithm.

[1] Supplementary models and information are available on http://www.tu-dresden.de/fghhihb/petzoldt/modlim/

After loading the required libraries, the model equations (the Holling equation `f` and the derivatives `model`), can be written very similar to the mathematical notation. To make the code more readable, the state vector `xx` is copied to named state variables and the parameters are extracted from the parameter vector `parms` using the `with`-environment. The results of the derivative function are returned as list.

```
library(odesolve)
library(scatterplot3d)

f <- function(x, y, k){x*y / (1+k*x)} #Holling II

model <- function(t, xx, parms) {
    u      <- xx[1]
    v      <- xx[2]
    w      <- xx[3]
    with(as.list(parms),{
    du <-  a * u  - alpha1 * f(u, v, k1)
    dv <- -b * v           + alpha1 * f(u, v, k1) +
                           - alpha2 * f(v, w, k2)
    dw <- -c * (w - wstar) + alpha2 * f(v, w, k2)
    list(c(du, dv, dw))
  })
}
```

As a next step, three vectors have to be defined: the `times` vector for which an output value is requested, the parameter vector (`parms`), and the start values of the state variables (`xstart`) where the names within the vectors correspond to the names used within the `model` function.

```
times  <- seq(0, 200, 0.1)
parms  <- c(a=1, b=1, c=10,
            alpha1=0.2, alpha2=1,
            k1=0.05, k2=0, wstar=0.006)
xstart <- c(u=10, v=5, w=0.1)
```

Now the simulation can be run using either `rk4` or `lsoda`:

```
out <- as.data.frame(lsoda(xstart, times,
                           model, parms))
```

This should take only two seconds on a standard computer and finally we can plot the simulation results as time series or state trajectory (fig. 1):

```
par(mfrow=c(2,2))
plot(times, out$u, type="l", col="green")
lines(times, out$v, type="l", col="blue")
plot(times, out$w, type="l", col="red")
plot(out$w[-1], out$w[-length(out$w)], type="l")
scatterplot3d(out$u, out$v, out$w, type="l")
```

Of course, similar results can be obtained with any other simulation package. However, the great advantage using a matrix oriented language like R or Octave[2] is, that the core model can be easily integrated within additional simulation procedures. As an example a bifurcation (Feigenbaum)-diagram of the chaotic amplitude of the predator population can be computed. First a small helper function which identifies the peaks (maxima and minima) of the time series is needed. Within a vectorized environment this can be implemented very easily by selecting all those values which are greater (resp. lower for minima) as their immediate left and right neighbours:

```
peaks <- function(x) {
  l <- length(x)
  xm1 <- c(x[-1], x[l])
  xp1 <- c(x[1], x[-l])
  x[x > xm1 & x > xp1 | x < xm1 & x < xp1]
}
```

As a next step the integration procedure is included within a main loop which increments the variable parameter `b`, evaluates the peaks and updates the plot:

```
plot(0,0, xlim=c(0,2), ylim=c(0,1.5),
          type="n", xlab="b", ylab="w")
for (b in seq(0.02,1.8,0.02)) {
  parms["b"] <- b
  out <- as.data.frame(lsoda(xstart, times,
                       model, parms, hmax=0.1))

  l <- length(out$w) %/% 3
  out <- out[(2*l):(3*l),]
  p <- peaks(out$w)
  l <- length(out$w)
  xstart <- c(u=out$u[l], v=out$v[l], w=out$w[l])
  points(rep(b, length(p)), p, pch=".")
}
```

After a stabilization phase only the last third of the detected peaks are plotted to show the behavior "at the end" of the time series. The resulting bifurcation diagram (fig. 2) shows the dependence of the amplitude of the predator w in dependence of the prey parameter b which can be interpreted as emigration parameter or as predator independent mortality. The interpretation of this diagram is explained in detail by Blasius and Stone (2000), but from the technical viewpoint this simulation shows, that within the matrix-oriented language R the integration of a simulation model within an analysis environment can be done with a very small effort of additional code. Depending on the resolution of the bifurcation diagram the cpu time seems to be relatively high (several minutes up to one hour) but considering that in the past such computations were often done on supercomputers, it should be acceptable on a personal computer.

[2] http://www.octave.org

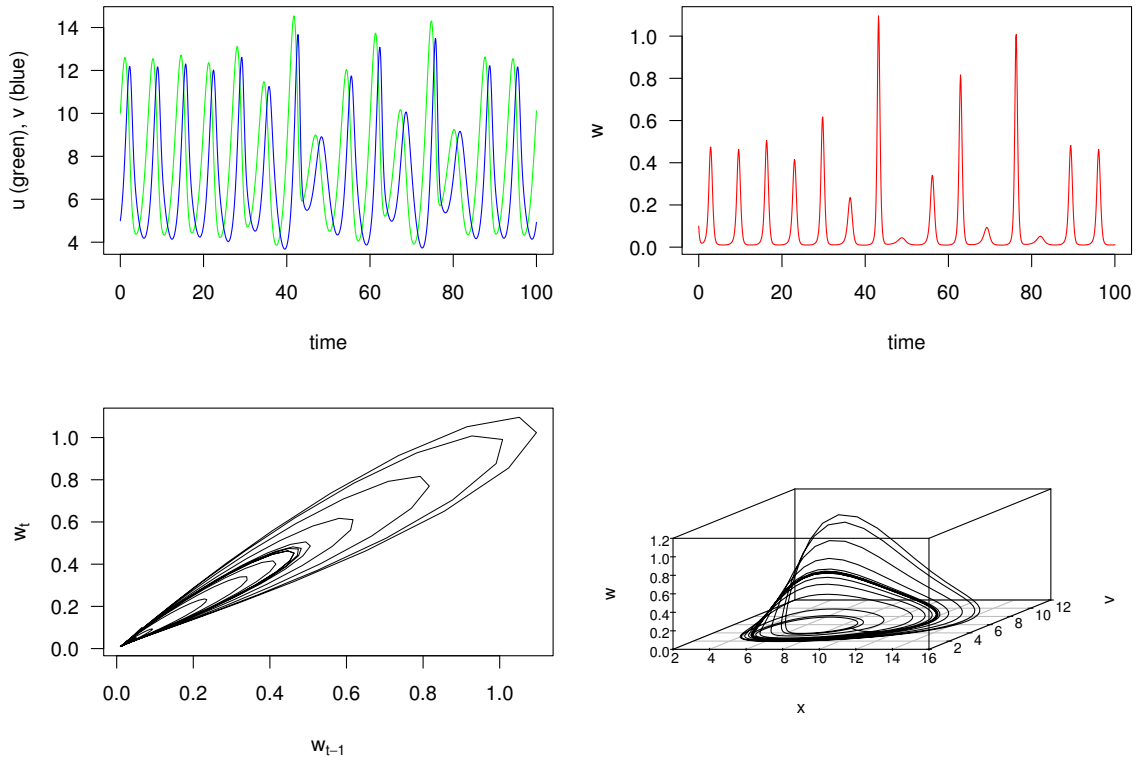

Figure 1: Simulation of an UPCA model, top left: resource (green) and prey (blue), top right: predator, bottom left: predator with lagged coordinates, bottom right: three dimensional state trajectory showing the chaotic attractor.

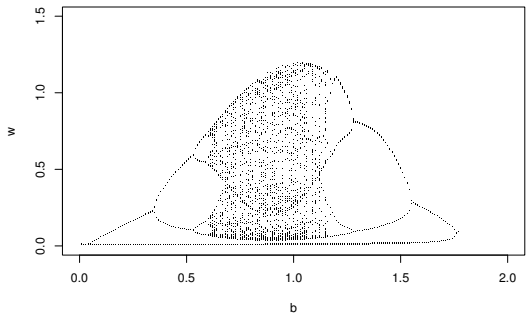

Figure 2: Bifurcation diagram for the predator w in dependence of prey parameter b.

Individual-based models

In contrast to ODE models, which in most cases work on an aggregated population level (abundance, biomass, concentration), individual based models are used to derive population parameters from the behavior of single individuals, which are commonly a stochastic sample of a given population. During the last decade this technique, which is in fact a creative collection of discrete event models, became widely used in theoretical and applied ecology (DeAngelis and Gross, 1992, and many others). Among them are very different approaches, which are spatially aggregated, spatially discrete (grid-based or cellular automata), or with a continuous space representation (particle diffusion type models).

Up to now there are different simulation tools available, which are mostly applicable to a relatively small class of individual-based approaches, e.g. SARCASim[3] and EcoBeaker[4] for cellular automata or simulation frameworks like OSIRIS (Mooij and Boersma, 1996). However, because the ecological creativity is higher than the functionality of some of these tools, a large proportion of individual-based models is implemented using general-purpose languages like C++ or Delphi, which in most cases, requires the programmer to take care of data input and output, random number generators and graphical visualization routines.

A possible solution of this tradeoff are matrix oriented languages like Matlab[5] (for individual-based models see Roughgarden, 1998), Octave or the S language. Within the R system a large collection of fundamental statistical, graphical and data manipu-

[3] http://www.collidoscope.com/ca/
[4] http://www.ecobeaker.com/
[5] http://www.mathworks.com/

lation functions is readily available and therefore it is not very difficult to implement individual-based models.

Individual-based population dynamics

The population dynamics of a *Daphnia* (water flee) population is shown here as an example. *Daphnia* plays an important role for the water quality of lakes (e.g. Benndorf et al., 2001), can be held very easily in the laboratory and the transparent skin makes it possible to observe important life functions using the microscope (e.g. food in the stomach or egg numbers in the brood pouch), so the genus *Daphnia* became an outstanding model organism in limnology, comparable to *Drosophila* in genetic research.

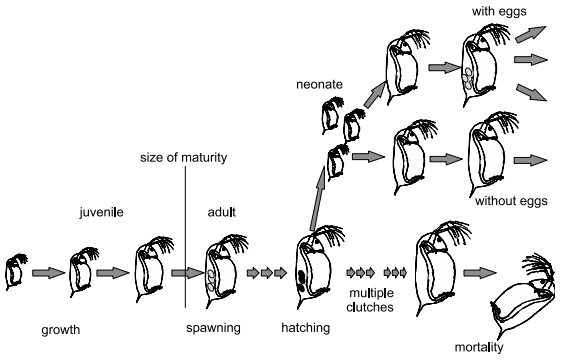

Figure 3: Parthenogenetic life cycle of *Daphnia*.

Under normal circumstances the life cycle of *Daphnia* is asexual (parthenogenetic) and begins with one new-born (neonate) female which grows up to the size of maturity (fig. 3). Then a number of asexual eggs are deposited into the brood pouch (spawning) which then grow up to neonates and which are released after a temperature-dependent time span into the free water (hatching). There are several approaches to model this life cycle in the literature (e.g. Kooijman, 1986; Gurney et al., 1990; Rinke and Petzoldt, 2003). Although the following is an absolutely elementary one which neglects food availability and simplifies size-dependent processes, it should be sufficient to demonstrate the principal methodology.

For the model we take the following assumptions (specified values taken from Hülsmann and Weiler (2000) and Hülsmann (2000)):

- The egg depelopment time is a function of water temperature according to Bottrell et al. (1976).
- Neonates have a size of 510 μm and the size of maturity is assumed to be 1250 μm.
- Juvenile length growth is linear with a growth rate of 83 $\mu m\, d^{-1}$ and due to the reproductive expense the adult growth rate is reduced to 80% of the juvenile value.
- The clutch size of each individual is taken randomly from an observed probability distribution of the population in Bautzen Reservoir (Germany) during spring 1999.
- The mortality rate is set to a fixed size-independent value of 0.02 d^{-1} and the maximum age is set to a fixed value of 30 d.

The implementation consists of six parts. After defining some constants, the vector with the cumulative distribution of the observed egg-frequencies, the simulation control parameters (1) and necessary helper functions, namely a function for an inverse sampling of empiric distributions (2), the life functions (3) of *Daphnia*: `grow`, `hatch`, `spawn` and `die` are implemented. Each individual is represented as one row of a data frame `inds` where the life functions either update single values of the individuals or add rows via `rbind` for each newborn individual (in `hatch`) or delete rows via `subset` for dead individuals. With the exception of the `hatch` function all procedures are possible as vectorized operations without the need of loops.

Now the population data frame is created (4) with some start individuals either as a fixed start population (as in the example) or generated randomly. The life loop (5) calls each life function for every time step and, as the `inds` data frame contains only one actual state, collects the desired outputs during the simulation, which are analysed graphically or numerically immediately after the simulation (6, see fig. 4). The example graphics show selected population statistics: the time series of abundance with an approximately exponential population growth, the mean length of the individuals as a function of time (indicating the presence of synchronized cohorts), the frequency distribution of egg numbers of adult individuals and a boxplot of the age distribution.

Additionally or as an alternative it is possible to save logfiles or snapshot graphics to disk and to analyse and visualise them with external programs.

```
#=============================================
# (1) global parameters
#=============================================
# cumulative egg frequency distribution
eggfreq     <- c(0.39, 0.49, 0.61, 0.74,
                 0.86, 0.95, 0.99, 1.00)
son         <- 510           # um
primipara   <- 1250          # um
juvgrowth   <- 83            # um/d
adgrowth    <- 0.8*juvgrowth # um/d
mort        <- 0.02          # 1/d
temp        <- 15            # deg C
timestep    <- 1             # d
steps       <- 40
#
# template for one individual
newdaphnia <- data.frame(age = 0,
                         size = son,
```

```
                      eggs = 0,
                     eggage = 0)
#=============================================
# (2) helper functions
#=============================================
botrell <- function(temp) {
    exp(3.3956 + 0.2193 *
       log(temp)-0.3414 * log(temp)^2)
}

# inverse sampling from an empiric distribution
clutchsize <- function(nn) {
    approx(c(0,eggfreq), 0:(length(eggfreq)),
           runif(nn), method="constant", f=0)$y
}
#=============================================
# (3) life methods of the individuals
#=============================================
grow <- function(inds){
    ninds       <- length(inds$age)
    inds$age    <- inds$age + timestep
    inds$eggage <- ifelse(inds$size > primipara,
                   inds$eggage + timestep, 0)
    inds$size   <- inds$size + timestep *
                   ifelse(inds$size < primipara,
                   juvgrowth, adgrowth)
    inds
}
die  <- function(inds) {
    subset(inds,
      runif(inds$age) > mort & inds$age <= 30)
}
spawn <- function(inds) {
  # individuals with size > primipara
  # and eggage==0 can spawn
  ninds          <- length(inds$age)
  neweggs        <- clutchsize(ninds)
  inds$eggs      <- ifelse(inds$size > primipara
                           & inds$eggage==0,
                           neweggs, inds$eggs)
  inds
}
hatch <- function(inds) {
    # individuals with eggs
    # and eggage > egg development time hatch
    ninds          <- length(inds$age)
    newinds        <- NULL
    edt <- botrell(temp)
    for (i in 1: ninds) {
        if (inds$eggage[i] > edt) {
          if (inds$eggs[i] > 0) {
            for (j in 1:inds$eggs[i]) {
                newinds <- rbind(newinds,
                             newdaphnia)
             }
          }
          inds$eggs[i]   <- 0
          inds$eggage[i] <- 0
        }
    }
    rbind(inds, newinds)
}
#=============================================
# (4) start individuals
#=============================================
inds <- data.frame(age  = c(7, 14, 1, 11, 8,
                            27, 2, 20, 7, 20),
                   size = c(1091, 1339, 618, 1286,
                            1153, 1557, 668, 1423,
                            1113, 1422),
                   eggs = c(0, 0, 0, 5, 0,
                            3, 0, 3, 0, 1),
                 eggage = c(0, 0, 0, 1.6, 0,
                            1.5, 0, 1.7, 0, 1.2))
#=============================================
# (5) life loop
#=============================================
sample.n        <- NULL
sample.size     <- NULL
sample.eggs     <- NULL
sample.agedist  <- NULL

for (k in 1:steps) {
    print(paste("timestep",k))
    inds <- grow(inds)
    inds <- die(inds)
    inds <- hatch(inds)
    inds <- spawn(inds)
    sample.n        <- c(sample.n,
                         length(inds$age))
    sample.size     <- c(sample.size,
                         mean(inds$size))
    sample.eggs     <- c(sample.eggs,
                         inds$eggs[inds$size >
                         primipara])
    sample.agedist  <- c(sample.agedist,
                         list(inds$age))
}
#=============================================
# (6) results and graphics
#=============================================
par(mfrow=c(2,2))
plot(sample.n, xlab="time (d)",
     ylab="abundance", type="l")
plot(sample.size, xlab="time (d)",
     ylab="mean body length (ţm)", type="l")
hist(sample.eggs, freq=FALSE, breaks=0:10,
     right=FALSE, ylab="rel. freq.",
     xlab="egg number", main="")
time <- seq(1,steps,2)
boxplot(sample.agedist[time],
        names=as.character(time), xlab="time (d)",
        ylab="age distribution (d)")
```

Particle diffusion models

In individual based modelling two different approaches are used to represent spatial movement: cellular automata and continuous coordinates (diffusion models). Whereas in the first case movement is realized by a state-transition of cells depending on their neighbourhood, in diffusion models the individuals are considered as particles with x and y coordinates which are updated according to a correlated or a random walk. The movement rules use polar coordinates with angle (α) and distance (r) which can be converted to cartesian coordinates using complex

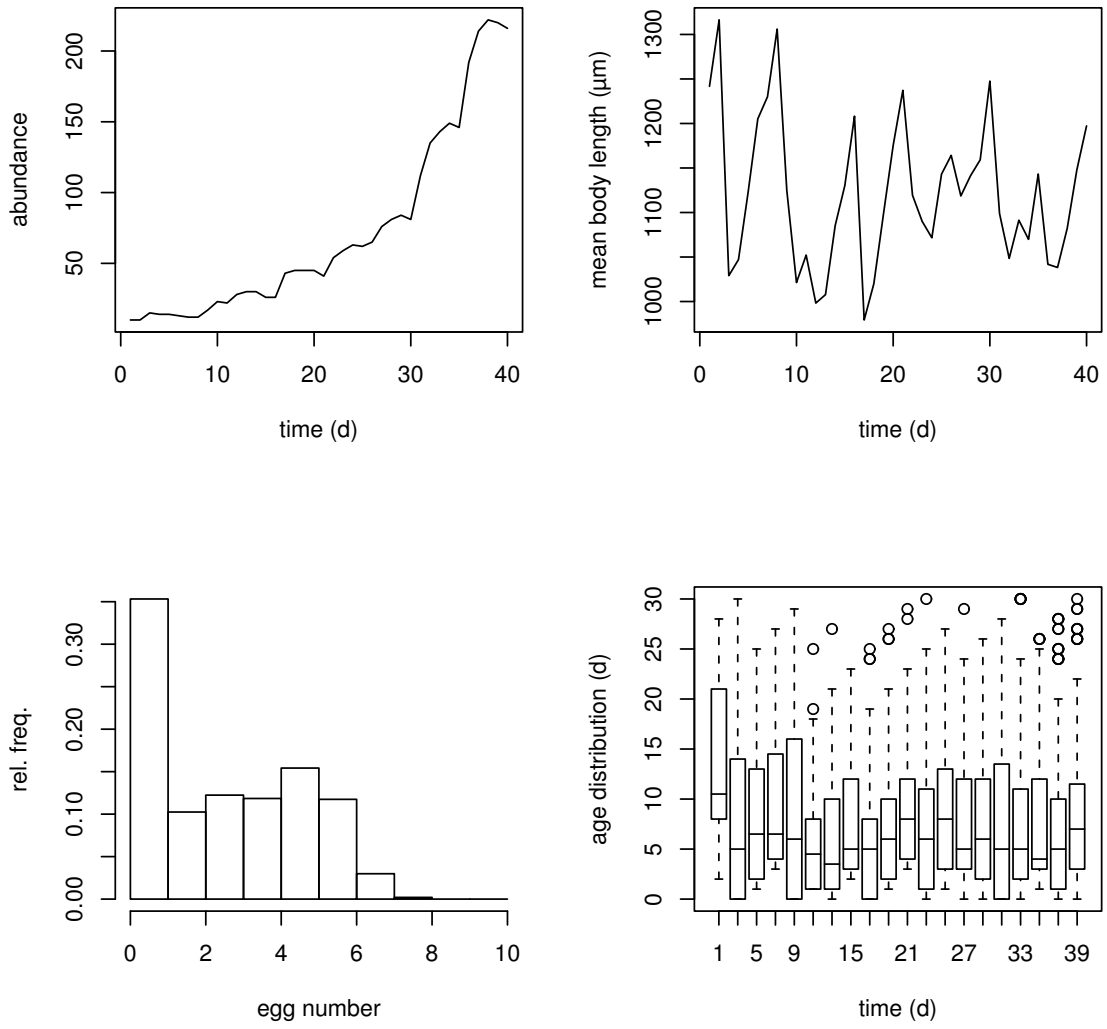

Figure 4: Results of an individual-based *Daphnia* model: exponential growth of abundance (top left), fluctuation of the mean body length indicating synchronised hatching (top right), simulated egg counts of adult individuals (bottom left), boxplot of age distribution as a function of time (bottom right).

numbers or sine and cosine transformation, what is essentially the same. Changes in direction are realized by adding a randomly selected turning angle to the angle of the time step before. An angle which is uniformly distributed within the interval $(0, 2\pi)$ results in an uncorrelated random walk, whereas an angle of less than a full circle or a non-uniform (e.g. a normal) distribution leads to a correlated random walk. To keep the individuals within the simulation area either a wrap around mode or reflection at the borders can be used.

The example shows a diffusion example with a coordinate system of $x, y = (0, 100)$ and an area of decreased movement speed in the middle $y = (45, 55)$, which can be interpreted as refugium (e.g. a hedge within a field for beatles), a region of increased food availability (the movement speed of herbivorous animals is less while grazing than foraging) or as a diffusion barrier (e.g. reduced eddy diffusion within the thermocline of stratified lakes).

The implementation is based on a data frame of individuals (`inds`) with cartesian coordinates (`x,y`) and a movement vector given as polar coordinates (`a, r`), the movement rules and the simulation loop.

The simulation runs fast enough for a real-time visualization of several hundred particles and gives an impression, how an increased abundance within refugia or diffusion barriers, observed very often in the field, can be explained without any complicated or intelligent behavior, solely on the basis of random walk and variable movement speed (fig. 5). This model can be extended in several ways and may include directed movement (sedimentation), reproduction or predator-prey interactions[6]. Furthermore, the principle of random walk can be used as a starting point for more complex theoretical models or, combined with population dynamics or bioenergetical models, as models of real-world systems (e.g. Hölker et al., 2002).

```
#=============================================
# simulation parameters and start individuals
#=============================================
n <- m <- 100  # size of simulation area
nruns  <- 2000 # number of simulation runs
nind   <- 100  # number of inds
inds   <- data.frame(x = runif(nind)*n,
                     y = runif(nind)*m,
                     r = rnorm(nind),
                     a = runif(nind)*2*pi)
#=============================================
# Movement
#=============================================
move <- function(inds) {
  with(inds, {
      ## Rule 1: Refugium
      speed <- ifelse(inds$y > 45
                   & inds$y < 55, 0.2, 2)
      ## Rule 2: Random Walk
      inds$a <- a + 2 * pi / runif(a)
      dx <- speed * r * cos(a)
      dy <- speed * r * sin(a)
      x<-inds$x + dx
      y<-inds$y + dy
      ## Rule 3: Wrap Around
      x<-ifelse(x>n,0,x)
      y<-ifelse(y>m,0,y)
      inds$x<-ifelse(x<0,n,x)
      inds$y<-ifelse(y<0,m,y)
      inds
  })
}
#=============================================
# main loop
#=============================================
for(i in 1:nruns) {
  inds <- move(inds)
  plot(inds$x, inds$y, col="red", pch=16,
     xlim=c(0, n), ylim=c(0, m),
     axes=FALSE, xlab="", ylab="")
}
```

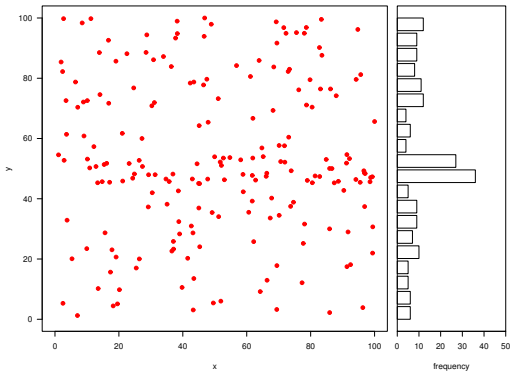

Figure 5: State of the particle diffusion model after 2000 time steps.

Cellular automata

Cellular automata (CA) are an alternative approach for the representation of spatial processes, widely used in theoretical and applied ecological modelling. As one of the most fundamental systems the well-known deterministic CA "Conway's Game of Life" (Gardner, 1970) is often used as an impressive simulation to show, that simple rules can lead to complex behavior. To implement a rectangular CA within R, a grid matrix (`z`), state transition rules implemented as `ifelse` statements and the `image` plot function are the essential building blocks. Furthermore a function (`neighbourhood`) is needed, which counts the number of active (nonzero) cells in the neighbourhood of each cell. In the strict sense only the eight adjacent cells are considered as neighbours, but in a more general sense a weighted neighbourhood matrix can be used. The neighbourhood count has to be evaluated for each cell in each time step and is therefore a computation intensive part, so the implementation as an

[6] see supplementary information

external C-function was necessary and is now available as part of the experimental **simecol** package.

Using this, Conway's Game of Life can be implemented with a very short piece of code. Part (1) defines the grid matrix with a set of predefined or random figures respectively, and part (2) is the life loop, which contains only the call of the neighbourhood function, the state transition rules and the visualization. The simulation is much slower than specialized Conway-programs (20 s on an AMD Athlon 1800+ for the example below) but still fast enough for real-time animation (0.2 s per time step). Approximately 85% of the elapsed time is needed for the visualization, only 3% for the neighbourhood function and the rest for the state transition rules.

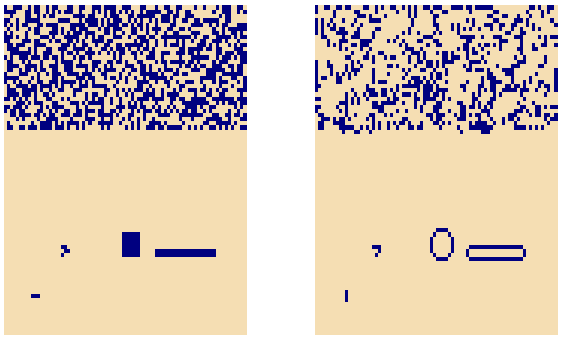

Figure 6: Initial state of matrix z before the simulation (left) and state after the first simulation step (right) .

However, despite the slower simulation speed it is a great advantage of this implementation, that state matrices and transition rules are under full control of the user and that the system is completely open for extension towards grid-based models with an ecological background. Furthermore the `neighbours`-function of the **simecol**-package allows the use of a generalized neighbourhood via a weight matrix to implement e.g. random or directed movement of animals, seed dispersal, forest fire propagation or epidemic models.

```
library(simecol)
#================================================
# (1) start individuals
#================================================
n <- m <- 80
z <- matrix(0, nrow=n, ncol=m)
z[40:45,20:25] <-1 # filled square 6x6 cells
z[51:70,20:21] <-1 # small rectangle 20 x 2 cells
z[10:12,10]    <-1 # short bar 3x1 cells
z[20:22,20:22] <- c(1,0,0,0,1,1,1,1,0) # glider
z[1:80,51:80]  <-round(runif(2400))    # random
image(z, col=c("wheat", "navy"), axes=FALSE)
#================================================
# (2) life loop
#================================================
for (i in 1:100){
  nb <- eightneighbours(z)
  ## survive with 2 or 3 neighbours
  zsurv <- ifelse(z > 0 & (nb == 2 | nb ==3), 1, 0)
  ## generate for empty cells with 3 neigbors
  zgen <- ifelse(z == 0 & nb == 3, 1, 0)
  z  <- matrix((zgen + zsurv), nrow=n)
  image(z, col=c("wheat","navy"),
        axes=FALSE, add=TRUE)
}
```

Conclusions

The examples described so far, plus the experience with R as data analysis environment for measurement and simulation data, allows to conclude that R is not only a great tool for statistical analysis and data processing but also a general-purpose simulation environment, both in research and especially in teaching.

It is not only an advantage to replace a lot of different tools on the modelers PC for different types of models, statistical analysis and data management with one single platform. The main advantage lies in the synergistic effects, which result from the combination and integration of these simulation models together with powerful statistics and publication quality graphics.

Being an interpreted language with a clear syntax and very powerful functions, R is easier to learn than other languages and allows rapid prototyping and and interactive work with the simulation models. Due to the ability to use vectorized functions, the execution speed is fast enough for non-trivial simulations and for real-time animation of teaching models. In many cases the code is compact enough to demonstrate field ecologists the full ecological functionality of the models. Furthermore, within a workgroup where field ecologists and modellers use the same computing environment for either statistical data analysis or the construction of simulation models, R becomes a communication aid to discuss ecological principles and to exchange ideas.

Acknowledgements

I am grateful to my collegues Karsten Rinke, Stephan Hülsmann, Robert Radke and Markus Wetzel for helpful discussions, to Woodrow Setzer for implementing the odesolve package and to the R Development Core Team for providing the great R system.

Bibliography

Benndorf, J., Kranich, J., Mehner, T., and Wagner, A. (2001). Temperature impact on the midsummer decline of *Daphnia galeata* from the biomanipulated Bautzen Reservoir (Germany). *Freshwater Biology*, 46:199–211. 11

Blasius, B., Huppert, A., and Stone, L. (1999). Complex dynamics and phase synchronization in

spatially extended ecological systems. *Nature*, 399:354–359. 8

Blasius, B. and Stone, L. (2000). Chaos and phase synchronization in ecological systems. *International Journal of Bifurcation and Chaos*, 10:2361–2380. 8, 9

Bottrell, H. H., Duncan, A., Gliwicz, Z. M., Grygierek, E., Herzig, A., Hillbricht-Ilkowska, A., Kurasawa, H., Larsson, P., and Weglenska, T. (1976). A review of some problems in zooplankton production studies. *Norvegian Journal of Zoology*, 24:419–456. 11

DeAngelis, D. L. and Gross, L. J., editors (1992). *Individual-Based Models and Approaches in Ecology: Populations, Communities, and Ecosystems.*, Knoxwille, Tennessee. Chapmann and Hall. Proceedings of a Symposium/Workshop. 10

Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game 'life'. *Scientific American*, 223:120–123. 14

Gurney, W. C., McCauley, E., Nisbet, R. M., and Murdoch, W. W. (1990). The physiological ecology of *Daphnia*: A dynamic model of growth and reproduction. *Ecology*, 71:716–732. 11

Hindmarsh, A. C. (1983). ODEPACK, a systematized collection of ODE solvers. In Stepleman, R., editor, *Scientific Computing*, pages 55–64. IMACS / North-Holland. 8

Hölker, F., Härtel, S. S., Steiner, S., and Mehner, T. (2002). Effects of piscivore-mediated habitat use on growth, diet and zooplankton consumption of roach: An individual-based modelling approach. *Freshwater Biology*, 47:2345–2358. 14

Hülsmann, S. (2000). *Population Dynamics of Daphnia galeata in the Biomanipulated Bautzen Reservoir: Life History Strategies Against Food Deficiency and Predation*. PhD thesis, TU Dresden Fakultät Forst-, Geo-, Hydrowissenschaften, Institut für Hydrobiologie. 11

Hülsmann, S. and Weiler, W. (2000). Adult, not juvenile mortality as a major reason for the midsummer decline of a *Daphnia* population. *Journal of Plankton Research*, 22(1):151–168. 11

Kooijman, S. (1986). Population dynamics on the basis of budgets. In Metz, J. and Diekman, O., editors, *The dynamics of physiologically structured populations*, pages 266–297. Springer Verlag, Berlin. 11

Mooij, W. M. and Boersma, M. (1996). An object oriented simulation framework for individual-based simulations (OSIRIS): *Daphnia* population dynamics as an example. *Ecol. Model.*, 93:139–153. 10

Petzold, L. R. (1983). Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *Siam J. Sci. Stat. Comput.*, 4:136–148. 8

Rinke, K. and Petzoldt, T. (2003). Modelling the effects of temperature and food on individual growth and reproduction of *Daphnia* and their consequences on the population level. *Limnologica*, 33(4):293–304. 11

Roughgarden, J. (1998). *Primer of Ecological Theory*. Prentice Hall. 10

Tierney, L. (2003). Name space management for R. *R News*, 3(1):2–6. 8

Thomas Petzoldt
Dresden University of Technology
Institute of Hydrobiology
Dresden, Germany
petzoldt@rcs.urz.tu-dresden.de