socialecological: Exploring socio-economics of arid rangelands

Florian D. Schneider

2016-07-14

Contents

This package vignette contains examples for the use of the functions provided in the package. It has the following main features:

1. provide a spatially explicit livestock resilience model for the cellular automata framework
2. initialise a list of landscape objects
3. update the list of landscape objects over a given period with a given set of parameters
4. summarize and analyse the distribution of the landscape states found in a list

the livestock model

The livestock model has been developed as a spatially explicit version of a graphical model deviced by Noy-Meir 1975[1] which explained alternative stable

[1]Noy-Meir, I. 1975. Stability of grazing systems: an application of predator-prey graphs. *Journal of Ecology* 63, 459-481.

states in rangelands from the logistic growth of plants and the non-linear feeding response of herbivore grazers. The intersections of plant growth and mortality define the potential steady states of the system.

The livestock model adds spatially explicit positive and negative interactions to the terms of growth and mortality which are based on assumptions of local facilitation and local competition, as well as plant defensive traits against grazing and attractant decoy effects.

While these assumptions define the shape of growth and mortality along the dual gradient of vegetation cover and spatial structure, the livestock rates and basic productivity of the system define the absolute extend of mortality and growth respectively.

There is a range of parameter combinations that allow for alternative stable states of the system, that is, the vegetation trends towards a different attractor if starting from high cover and clustering than it would at low cover and little clustering. The history of the ecosystem defines about the direction of its further development (Hysteresis).

The basins of attraction can be approached in a pair-approximation simplification of the model. However, only the stochastic noise of the cellular automata allows for an investigation of likelihood of future outcomes. Thus, by adding further stochastic elements to the model, we can assess uncertainties in terms of future revenue or success of a management method.

More details on the model can be found in the vignette example.

generate landscapes and run simulations using caspr

The caspr package provides a framework to run cellular automata simulations in R. Many of the functions provided in this example refer to functions of caspr. So first, both packages have to be loaded into your current R session.

```
library(socialecological)
library(caspr)
```

```
##
## Attaching package: 'caspr'
```

```
## Das folgende Objekt ist maskiert 'package:socialecological':
```

```
##
##     livestock
```

Now we can generate landscapes using the function `init_landscape()`:

```
l <- init_landscape(
          states= c("1","0"), # potential cell states: 1 = vegetated, 0 = empty g
          cover = c(0.7,0.3), # initial random cover
          width = 50          # width and height of the landscape object
          )
plot(l)
```

```
## Warning in plot.landscape(l): No colors provided for landscape plotting:
## choosing ad-hoc grey levels
```

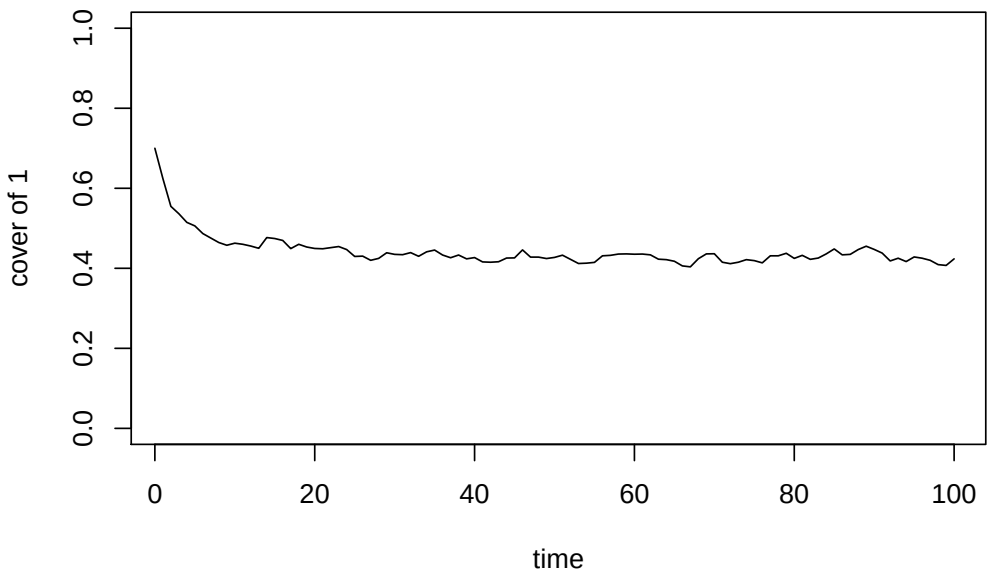
The function `ca()` is the core function of caspr and applies a cellular automata simulation to the given landscape with a set of parameters:

```
p <- list(
  r = 1.0,  # max. regeneration rate of plants
  b = 0.2,  # environmental quality
  sigma = 0, # environmental noise
  f = 0.9,  # local facilitation
  alpha = 0, # water runoff
  K = 0.9, # carrying capacity of the system
  c = 0.2, # local competition
  m = 0.05, # intrinsic mortality of plants (inverse of av. lifespan)
  v = 0.8, # attractant-decoy
  p = 0, # associational resistance
  L = 10, # Livestock density
  q = 0, # hill exponent of functional response
  h = 50, # handling time
  a = 0.2 # attack rate of livestock
)

r <- ca(l, model = livestock, parms = p, t_max = 100)
```

The plot function knows how to handle output of this model run. It automatically plots a timeseries of vegetation cover. Alternatively, single snapshots of landscapes can be assessed at any timestep.

```
plot(r) # plot timeseries of vegetation cover
```




```
par(mfrow = c(1,4), mar = c(1,1,1,1)) # plot snapshots of landscapes at certain t
plot(r$landscapes[[1]])
plot(r$landscapes[[25]])
plot(r$landscapes[[50]])
plot(r$landscapes[[100]])
```

add random environmental noise

The parameters `b` and `sigma` define the environmental quality and the random variability of it across years. Thus, if specifying sigma, the climate effects vary randomly over time. In the following example code, the longtime outcome of the simulation is only little affected by high variation in environmental quality (red case). However, in situations close to the tipping point, high stochasticity may be inducing the shift earlier.

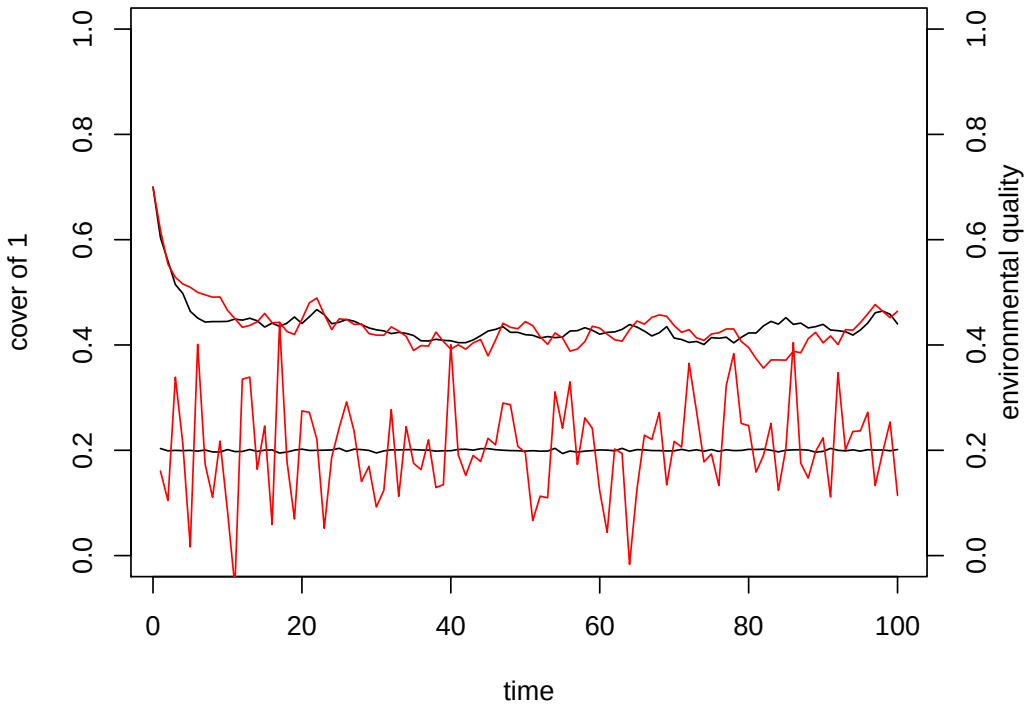
```
p$sigma = 0.01

r <- ca(l, model = livestock, parms = p, t_max = 100)
par(mar=c(4,4,1,4))
plot(r)
axis(4)
mtext("environmental quality", side = 4, line = 2)
temp <- with(p, b*rnorm(100, 1, sigma))
lines(temp)
```

```
p$sigma = 0.5

run <- ca(l, livestock, p, t_max = 100, saveeach = 100)
lines(run$time, run$cover[[1]], col = "red")

temp <- with(p, b*rnorm(100, 1, sigma))
lines(temp, col = "red")
```



set parameter time series

Manipulations to the parameters over time will be required for the intended research questions. Either because the questions address the impacts of ongoing climate change on arid landscape stability without intervention, or because the consequences of intervention by management methods are to be assessed. In both cases, a predefined time sequence of a parameter needs to be fed into the updating procedure of the cellular automata. This will be achieved by providing the model with vectors instead of single value parameters. Each vector has to have the length of the requested timeseries or it will be recycled, e.g. for rotational management measures.

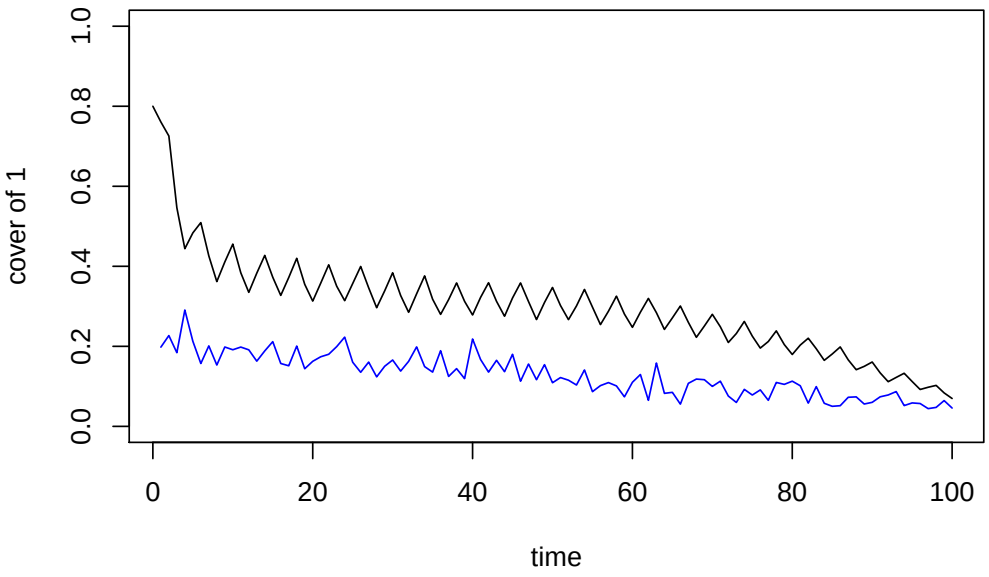
Thus, as required by the caspr package, the parameter set is given as a list of parameters, but each list entry can be either a single value or a vector representing the sequence of parameter values over time (in years). This is inflated by the requested time to evaluate using the function `parms_timeseries()` of caspr.

```
p <- list(
  r = 1.0,  # max. regeneration rate of plants
  b = seq(0.2,0.05, length = 100)*rnorm(100, 1, 0.2),  # environmental quality
  sigma = 0, # random annual variation of environmental quality
  f = 0.9,  # local facilitation
  alpha = 0, # water runoff
  K = 0.7, # carrying capacity of the system
  c = 0.2, # local competition
  m = 0.05, # intrinsic mortality of plants (inverse of av. lifespan)
  v = 0.2, # attractant-decoy
  p = 0.99, # associational resistance
  L = rep(c(0,15), each = 2), # Livestock density
  q = 0, # hill exponent of functional response
  h = 50, # handling time
  a = 1.0 # attack rate of livestock
)


plist <- parms_timeseries(p, 100)

l <- init_landscape(c("1","0"), cover = c(0.8,0.2), width = 100)

r <- ca(l, livestock, plist, t_max = 100, saveeach = 5)
plot(r)
lines(1:100, p$b, col = "blue")
```



lists of landscape objects

In a next step, we want to explore the space of possible outcomes under a given set of parameters. This is achieved by running the model iteratively, starting from steady state situations, and by mapping the trajectories and analysing the final state of the model after a couple of years.

timeseries simulations and steady state

I wrote a set of functions which take a list of landscape objects (created by function `init_list()`) and updates it with the given parameter set for the requested time period (by applying function `update_list()`). If a parallel backend is provided, the single runs are processesed in parallel. The output is an updated list of landscape objects which can be summarized into a report of the distribution of outcomes.

All simulations have to start out from a pre-formed landscape with realistic patch structure, since the spatial structure is what determines the future

development of the landscape. That is particularly important if looking at only short time spans such as 5 or 50 years. For now, we assume that steady state is achieved after a period of 100 years.

```
L0 <- init_list(100, runif_range = c(0.6,0.99), width = 25)

p <- list(
  r = 1.0,  # max. regeneration rate of plants
  b = 0.2,  # environmental quality
  sigma = 0.1, # random annual variation of environmental quality
  f = 0.9,  # local facilitation
  alpha = 0, # water runoff
  K = 0.9, # carrying capacity of the system
  c = 0.2, # local competition
  m = 0.05, # intrinsic mortality of plants (inverse of av. lifespan)
  v = 0.0, # attractant-decoy
  p = 0.9, # associational resistance
  L = 20, # Livestock density
  q = 0, # hill exponent of functional response
  h = 30, # handling time
  a = 0.3 # attack rate of livestock
)
L100 <- update_list(L0, 100, p)

summary(L100)
```

```
## Assessing n = 100 landscapes:
##   mean total cover:   0.0387 (Ã± 0.0102 )
##   mean local cover:   0.0719 (Ã± 0.0372 )
##   clustering coefficient:     1.92
```

Now, we can think of many different scenarios, e.g. the application of a management method for restoration. Let's say after a long period of intensive grazing, the landscape is to be grazed only by one tenth of the livestock rates.

```
p$b <- 0.1
p$L <- 2

L110 <- update_list(L100, 10, p)
summary(L110)
```

```
## Assessing n = 100 landscapes:
##   mean total cover:   0.0731 (Ã± 0.0324 )
##   mean local cover:   0.1498 (Ã± 0.0664 )
##   clustering coefficient:     2.1
```

```
L120 <- update_list(L110, 10, p)
summary(L120)
```

```
## Assessing n = 100 landscapes:
##   mean total cover:   0.1615 (Ã± 0.1006 )
##   mean local cover:   0.2661 (Ã± 0.1218 )
##   clustering coefficient:     1.8
```

visualisation and analysis

Violin plots represent the distribution of the iterations current cover and can be interpreted as probabilities. The distribution pattern varies with the number of replicates, the amount of environmental noise and size of the landscape.

```
library(vioplot)
```

```
## Loading required package: sm
```

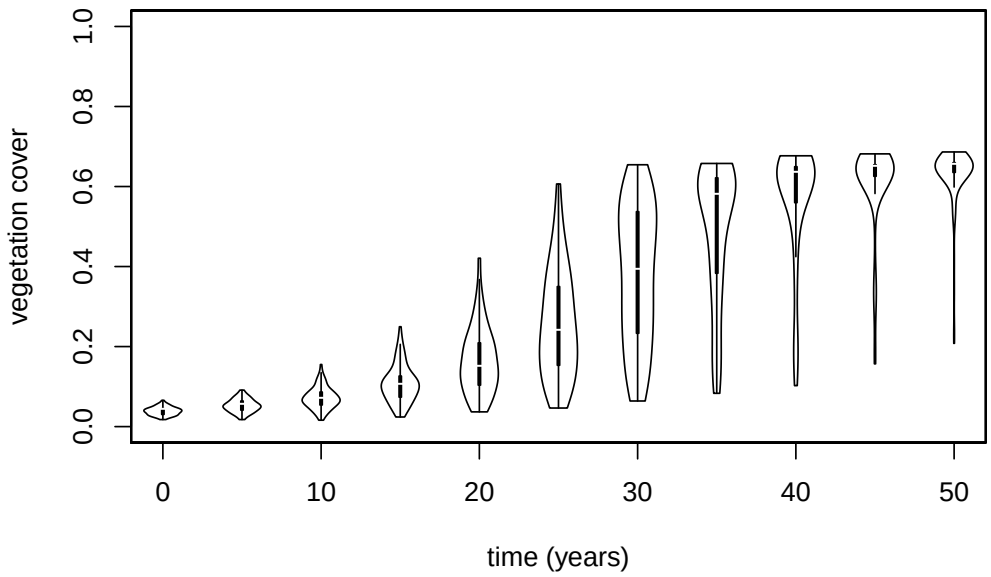
```
## Package 'sm', version 2.2-5.4: type help(sm) for summary information
```

```
plot(NA,NA, xlim = c(0,50), ylim  = c(0,1),
     xlab = "time (years)", ylab = "vegetation cover" )
vioplot(sapply(L100, function(l) summary(l)$cover[1]),
        at = 0, add = T,
        col = 'white', wex = 3, pchMed = "-", colMed = "white")

# continue running in steps of 5 years but at increased environmental quality

L <- L100
p$b <- 0.1
p$L <- 2
```

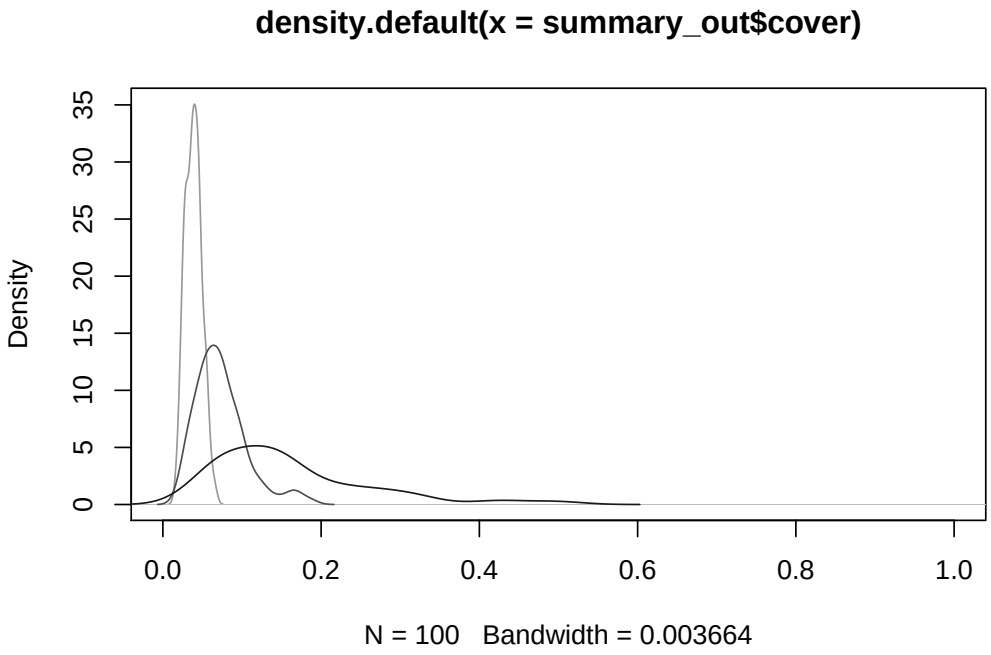
```
for(i in seq(5,50, 5)) {
L <- update_list(L, 5, p)
vioplot(sapply(L, function(l) summary(l)$cover[1]),
        at = i, add = T,
        col = 'white', wex = 3, pchMed = "-", colMed = "white")
}
```



The violin plot is a combination of a boxplot and a distribution kernel. The kernel itself is stored in the output created by the summary function and can be assessed for further analysis, e.g. to match it with economic revenue of a particular landscape state.

```
plot(summary(L100)$kernel, type = "l", xlim = c(0,1), col = "grey60")

lines(summary(L110)$kernel, type = "l", col = "grey30")
lines(summary(L120)$kernel, type = "l", col = "grey10")
```


density.default(x = summary_out$cover)
Density
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0
N = 100 Bandwidth = 0.003664