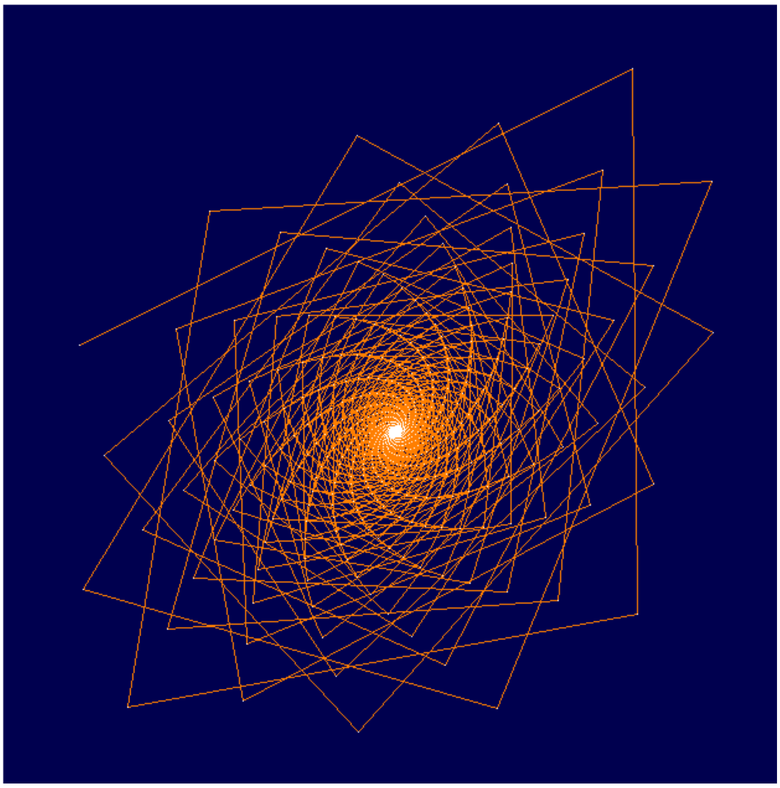
UNIFIED LIFE MODELS

ULM: Reference Manual

Population Dynamics
Version 6.0



https://www.biologie.ens.fr/~legendre/ulm/ulm.html
https://gitlab.com/ecoevomath/ulm

CONTENTS

1 FIRST STEPS

1.1 Contents of ULM distribution

`ulm.exe` or `ulm`	ULM program (Windows® and Linux/macOS® respectively)
`ulmc.exe` or `ulmc`	console (no graphics) version
`ulmref.pdf`	reference manual

1.2 Run

The ULM program should have been installed from the compressed archive (`ulm.zip` under Windows®, `ulm.tar.gz` under Linux, `ulm.dmg` under macOS®) in a directory `ulm`.

- Open the ULM program.
- From the ULM main window, use the option *File* | *New from example* or the button 🗎 to load an example model (the list of models is provided in Table 1.1). This opens the *Model files* window where the model is displayed. Models can be edited in this window.
- Click the *Compile* button 💡. Click the *Run* button ▶.

To develop your own model you can either:

- Load an example model that is close to your model of interest. In this way, you shall get used to the ULM syntax. Modify the example model using the editing facilities of the *Model files* window, save it with the extension `ulm` (e.g. `my_model.ulm`).
- You can alternatively use a text editor to create your model file.

To stop execution of the program, use the *Stop* button ⊗.

1.3 Example models provided with ULM

See Table 1.1 for the list of all example models provided with ULM. Those models can be loaded using the *New from example* button 🗎. If you are compiling ULM from source, they are located in the `models/` directory. Most models follow the following convention:

- Suffixed with 0: constant matrix model (e.g. `droso_0.ulm`)
- Suffixed with 2: 2-sex model (e.g. `bigh_02.ulm`)
- Suffixed with m: multisite model (e.g. `gull_0m.ulm`)
- Suffixed with g: size-classified model (e.g. `kwha_0g.ulm`)
- Suffixed with d: density-dependent model (e.g. `droso_d.ulm`)
- Suffixed with e: model with environmental stochasticity (e.g. `bigh_2se.ulm`)
- Suffixed with s: model with demographic stochasticity (e.g. `pass_2s.ulm`)

`allee.ulm`	probability of mating
`astrocaryum_mexicanum_0.ulm`	complex life cycle for the palm tree *Astrocaryum mexicanum*
`bigh_02.ulm`	2 sex life cycle for Bighorn sheep *Ovis canadensis*
`bigh_2se.ulm`	stochastic 2 sex model for Bighorn sheep
`curv.ulm`	display various curves in the plane
`dipsacus_sylvestris_0`	complex life cycle for the teasel *Dipsacus sylvestris*
`droso_0.ulm`	life cycle for the fruit fly *Drosophila melanogaster*
`droso_d.ulm`	model with density dependence and contamination for *Drosophila melanogaster*
`griza_0.ulm`	life cycle for the Grizzly bear *Ursus arctos horribilis* (increasing population)
`griza_2s.ulm`	2-sex life cycle with demographic stochasticity for Grizzly bear
`griza_2sd.ulm`	2-sex life cycle with demographic stochasticity and density dependence for Grizzly bear
`griza_2se.ulm`	2-sex life cycle with demographic and environmental stochasticity for Grizzly bear
`grizb_0.ulm`	life cycle for Grizzly bear *Ursus arctos horribilis* (declining population)
`grizb_2s.ulm`	2-sex life cycle with demographic stochasticity for Grizzly bear
`grizb_2sd.ulm`	2-sex life cycle with demographic stochasticity and density dependence for Grizzly bear
`grizb_2se.ulm`	2-sex life cycle with demographic and environmental stochasticity for Grizzly bear
`gull_0m.ulm`	multisite model for Black-headed gull
`henon.ulm`	Hénon attractor
`henon.ulm`	Hénon attractor
`kwha_0g.ulm`	size-classified life cycle for Killer whale
`kwha_sg.ulm`	model with demographic stochasticity for Killer whale
`lorenz.ulm`	Lorenz attractor
`met_0.ulm`	life cycle for the spider *Metepeira datona*
`met_esd.ulm`	extinction dynamics for the spider *Metepeira datona*
`pass_0.ulm`	generic life cycle for passerine
`pass_02.ulm`	2-sex life cycle for passerine
`pass_2s.ulm`	2-sex model with demographic stochasticity for passerine
`passa_0.ulm`	post-breeding census life cycle for passerine
`qsd_sd.ulm`	study of quasi-stationary distribution
`regis.ulm`	density dependence with chaotic dynamics
`regis.in`	command file for model file `regis.ulm`
`snowg_0.ulm`	life cycle for the snow goose *Chen caerulescens atlantica*
`usa_0.ulm`	life cycle for the USA population
`variation_extinction.ulm`	study of extinction and recolonization events
`vult_0.ulm`	life cycle for the griffon vulture *Gyps fulvus*
`vulta_0.ulm`	release strategies for vulture populations
`vultb_0.ulm`	release strategies for vulture populations
`vult_2se.ulm`	2-sex model with demographic and environmental stochasticity for vulture

Table 1.1: Example models.

2 UNIFIED LIFE MODELS

The ULM computer program has been designed to study population dynamics, with applications to evolutionary and conservation biology. Matrix population models, and deterministic or stochastic relations in discrete time can be explored interactively by means of simple commands and convenient graphics.

Density dependence, environmental stochasticity, demographic stochasticity, and migrations can be taken into account. Some theoretical knowledge about matrix population models and population dynamics [Caswell, 1989, Caswell, 2001] is required to build model files, but the ULM program is easy to use.

The biological system under study is described in a text file, the model file, using a reduced declaration language, and appropriate mathematical functions. The ULM program is run with the model file as input. The model can then be studied via population trajectories, matrix properties, sensitivities, Monte Carlo simulations. The ULM kernel is a symbolic evaluator.

The main point of ULM is that population dynamics can be modeled finely, with plain knowledge of what is simulated, and without heavy programming.

Author

Stéphane Legendre
Team of Mathematical Eco-Evolution
École Normale Supérieure
46 rue d'Ulm
75000 Paris – France
legendre@ens.fr

Contributors Guillaume Chapron, Jean Clobert, Régis Ferrière, Frédéric Gosselin, Jean-Dominique Lebreton, François Sarrazin, Karl-Michael Schindler, Alexis Simon.

ULM development team François Bienvenu, Guilhem Doulcier, Hugo Gruson, Maxime Woringer.

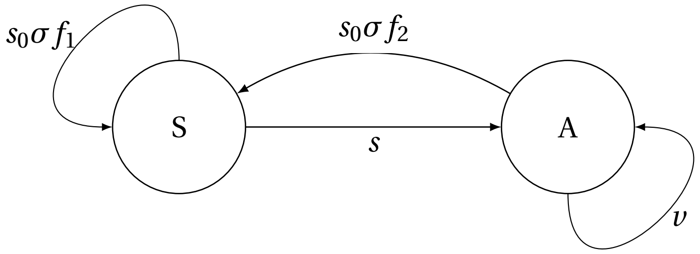
Distribution The ULM computer program is distributed as a free/open source software under the GNU General Public License version 3 (GNU GPLv3). Users are under their own responsibility.

3 PROJECTING POPULATIONS

3.1 The life cycle

The first step for studying population dynamics of a species in discrete time is to build the life cycle. The life cycle is a macroscopic description of an organism within a population, by means of stages and transitions between stages. It incorporates the genotype and part of the phenotype.

We use a generic model for passerine as example [Legendre et al., 1999]. The model is female-based, with 2 age classes. It is assumed that survival rates are computed from counts of individuals just before reproduction (pre-breeding census). The first age class is constituted of subadults, aged (almost) one year. The second age class is constituted of adults, aged 2 years and more. Subadult females reproduce with fecundity f_1, and adult females reproduce with fecundity f_2.



3.2 The population matrix

Let $n_1(t)$ the number of subadults at time t, and $n_2(t)$ the number of adults at time t. Then:

$$n_1(t+1) = s_0 \sigma f_1 n_1(t) + s_0 \sigma f_2 n_2(t).$$

Indeed, at time $t+1$, $n_1(t+1)$ individuals aged one year are born from $n_1(t)$ subadults at time t, and $n_2(t)$ adults at time t. Newborn individuals survived with juvenile survival rate s_0. To take account of females only, the numbers are multiplied by the primary female sex ratio $\sigma (\sigma = 0.5)$. Similarly:

$$n_2(t+1) = s n_1(t) + v n_2(t)$$

as the number $n_2(t+1)$ of adults at time $t+1$ is the number of subadults that survived with rate s, plus the number of adults that survived with rate v.

These two relations can be put in matrix form:

$$\begin{bmatrix} n_1 \\ n_2 \end{bmatrix}_{t+1} = \begin{bmatrix} s_0 \sigma f_1 & s_0 \sigma f_2 \\ s & v \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}_t ,$$

or

$$N(t+1) = AN(t),$$

with $N(t)$ the population vector at time t, and A the (constant) population matrix. Population size at time t is the sum of the vector entries: $n(t) = n_1(t) + n_2(t)$.

From these equations, the syntax of the corresponding ULM example model `pass_0` is straightforward. ULM keywords are written in bold, comments are added on the right. Note that declarations are separated by blank lines.

```
defmod passerine(2)           model declaration, 2 age classes
mat: a                        name of population matrix
vec: w                        name of population vector

defvec w(2)                   population vector
n1, n2

defmat a(2)                   population matrix
sigma*s0*f1, sigma*s0*f2
s, v

defvar n1 = 10                initial number of subadults

defvar n2 = 10                initial number of adults

defvar n = n1 + n2            total population size

defvar s0 = 0.2               juvenile survival rate

defvar s = 0.35               subadult survival rate

defvar v = 0.5                adult survival rate

defvar sigma = 0.5            primary female sex ratio

defvar f1 = 7                 subadult female fecundity

defvar f2 = 7                 adult female fecundity
```

The declaration language is described in Section 4. Load the example passerine model `passa_0` (using the *New from example* button 🗎) for post-breeding census. The model has 3 age classes.

3.3 Running the model

- Load the example model `pass_0` using the *New from example* dialog 🗎. A new file containing the model appears in the *Model file* window. (Note: You can modify it and save your modifications in a `.ulm` file).

- Click the *Compile* button 💡. The file is processed (checked for syntactic errors), and can now be studied interactively.

- Click the *Run* button ▶. The model is run for 50 time steps (the default). Population trajectories are displayed in graphic window #1 (Figure 3.1). In the large panel of the main window, the number n_1 of subadults and the number n_2 of adults are displayed every 10 time steps.

```
> Run 50
  t = 10
    n1 = 35.06
    n2 = 20.28
  t = 20
    n1 = 95.14
    n2 = 55.04
  t = 30
    n1 = 258.2
    n2 = 149.4
  t = 40
    n1 = 700.5
    n2 = 405.3
  t = 50
    n1 = 1901
    n2 = 1100
  Model passerine -> pop = 3000.4
  growth rate from [t = 0] -> 1.105409
```

Starting from 20 individuals at time $t = 0$ (as specified in the model file), the population reaches 3000 individuals at time $t = 50$. The corresponding growth rate is 1.10.

- Click the *Settings* button ✥ in graphic window #1 to parameterize the graphics. Replace $n1$ by n (total population size) and remove $n2$. Click OK.

- Click the *Init* button ·. The system is initialized (t = 0). Select option *Run* | *Settings*. Set *Number of time steps* to 20 (instead of 50). Click OK.

- Click the *Run* button ▶. Total population size n is displayed in graphic window #1 for 20 time steps.

The previous commands could have been executed by typing in the small *Interpreter* panel of the main window:

? **graph** t n	*set graphics to display n along time t*
? **init**	*initialize (t = 0)*
? **run** 20	*run for 20 time steps*

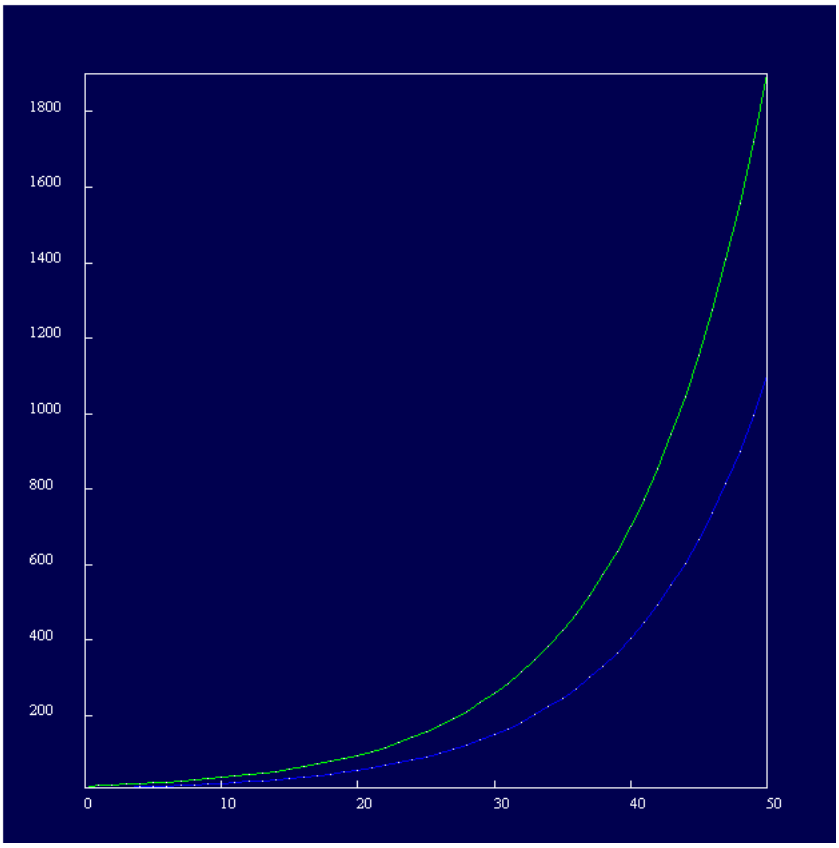


Figure 3.1: Subadult (green) and adult (blue) population sizes along time for the passerine model (model `pass_0`), showing exponential growth and stable age distribution.

3.4 Matrix properties

For constant matrix models like `pass_0`, population dynamics are entirely known from algebraic properties of the matrix.

- Click the *Matrix property* button P to get the *Matrix properties* window. The dominant eigenvalue *lambda* of the population matrix *a* is 1.10498.

Growth rate The dominant eigenvalue $\lambda = 1.10498$ is the long-term growth rate of the population. Indeed, for large t we have:

$$n(t+1) \sim \lambda n(t), \text{ and } n(t) \sim C\lambda^t n(0),$$

where C is a constant precised below. These equations mean that the population increases or decreases geometrically with rate λ. From the second equation an estimator of the growth rate is computed:

$$\hat{\lambda} = \exp\left[\frac{\ln(n(t)) - \ln(n(0))}{t}\right] \tag{3.1}$$

This estimator is displayed when the model is run (see previous section). For population matrices, which are non negative, the dominant eigenvalue exists in most cases.

- In the central panel of the *Matrix properties* window the right eigenvector $W = [0.63, 0.37]$ of the matrix with respect to the dominant eigenvalue λ is displayed.

Stable age distribution The right eigenvector W is the stable stage distribution. Let $W(t) = \left[\frac{n_1(t)}{n(t)}, \frac{n_2(t)}{n(t)}\right]$ be the population structure at time t, that is the proportion of individuals in stages. As t gets large, we have $W(t) \to W$. In an age-classified model, W is the stable age distribution. At demographic equilibrium, the passerine population is constituted of 63 % subadults and 37 % adults.

- In the small *Interpreter* panel type the following commands:

```
? newvar p1 n1/n   create new variable p1, proportion of individuals in age class 1
? newvar p2 n2/n   create new variable p2, proportion of individuals in age class 2
? graph t p1 p2    set graphics to display p1 and p2 along time t
? yscale 0 1       fix bounds on Y axis
? run 20           run for 20 time steps
```

This illustrates the convergence of age structure towards the stable age distribution.

- In the central panel of the *Matrix properties* window the left eigenvector $V = [0.46, 0.54]$ of the matrix with respect to the dominant eigenvalue λ is displayed.

Reproductive value The left eigenvector V is the reproductive value. It reflects the contribution of each class to population size.

- In the small *Interpreter* panel type the following commands:

```
? change n1 20                   change initial number of subadults to 20
? change n2 0                    change initial number of adults to 0
                                 total initial population size is still n(0) = 20
? run 50                         run for 50 time steps
Model passerine -> pop = 2782.0
? change n1 0                    change initial number of subadults to 0
? change n2 20                   change initial number of adults to 20
                                 total initial population size is still n(0) = 20
? run 50                         run for 50 time steps
Model passerine -> pop = 3218.9
```

This illustrates that introducing adults leads to a larger population size than introducing the same number of subadults. Adults have a larger reproductive value. The exact formula is:

$$n(t) \sim \langle V, W(0)\rangle \lambda^t n(0)$$

with the dot product $C = \langle V, W(0)\rangle$. The long-term population size depends on the initial population size $n(0)$, on the initial proportions of individuals in age classes (the initial population structure $W(0)$), and on the reproductive value V. The growth rate λ is independent of initial population size or structure.

3.5 The survival-fertility decomposition of the projection matrix

The projection matrix A associated with the life cycle graph of the species can be decomposed $A = S + F$. Here F is the matrix of *reproductive transitions*, those that lead to the production of offspring: the entry f_{ij} of F is the expected number of class i offspring at time $t+1$ produced by an individual in class j at time t. S is the matrix of *survival transitions*: the entry s_{ij} of S is the probability that an individual in stage j at time t is alive and in stage i at time $t+1$.

This decomposition allows to compute several demographic descriptors for a complex life cycle, one that is not age-classified, e.g. a size-classified one. The reproductive transitions can be specified by appending the character '#'. For example, the population matrix for the teasel *Dipsacus sylvestris* (load example model `dipsacus_sylvestris_0` using the button 🗎) is declared with 4 reproductive transitions:

```
defmat a(6)
0,       0,      0,      0,      0,      322.38#
0.966,   0,      0,      0,      0,      0
0.013,   0.010,  0.125,  0,      0,      3.448#
0.007,   0,      0.125,  0.238,  0,      30.17#
0.008,   0,      0.038,  0.245,  0.167,  0.862#
0,       0,      0,      0.023,  0.750,  0
```

3.6 Sensitivities

When a parameter x of the model is varied by an amount ϵ, the growth rate λ changes by an amount ϵS_x, where $S_x = \frac{\partial \lambda}{\partial x}$ is the sensitivity of λ to changes in x. When a parameter x of the model is varied by α%, the growth rate λ changes by αE_x%, where $E_x = \frac{x}{\lambda} S_x$ is the elasticity of λ to changes in x. Elasticity is similar to sensitivity, but takes the size of the parameter into account. Sensitivities and elasticities allow to determine which parameters have the greatest impact on population growth. This is important for population management and conservation, in relation with environmental impact on demographic parameters. Sensitivities also have an interpretation in terms of the selective value of phenotypic traits.

- Select option *Matrix* | *Sensitivities* to get the *Sensitivities* window where sensitivities and elasticities of λ to changes in matrix entries are displayed. Type s in the *Sensitivity to variable* panel, then `<return>`. The sensitivity of λ to changes in subadult survival rate s obtains $S_s = 0.6931$, and the elasticity obtains $E_s = 0.2195$.

The computation can be done for adult survival rate v, juvenile survival s_0, fecundities f_1, f_2. It is found that juvenile survival s_0 is by far the most sensitive parameter with $S_{s_0} = 3.309$, and $E_{s_0} = 0.599$, as is the case for short-lived bird species. The following formula holds [Houllier and Lebreton, 1986]: $E_{s_0} = \frac{1}{\bar{T}}$, with $\bar{T} = 1.669$ the mean generation length (see the *Matrix properties* window).

- Click button to display the list of variables. Click in the expression for s_0. Change the constant value 0.2 by 0.21, a 5 % change in s_0. Type `<return>`. Select the *Matrix properties* window. The value of *lambda* is updated to $\lambda' = 1.13813$. By the definition of elasticity E_{s_0}, a 5 % increase in s_0 induces a $0.05 \times 0.599 = 0.03 = 3\%$ increase in λ. It is checked that $\lambda' - \lambda = 1.13813 - 1.10498 = 0.03$.
- In the *Interpreter* panel, type the following to go back to the reference value 0.2 of juvenile survival s_0 (or use the button):

? **change** s0 0.2

- Select option *Matrix* | *Landscape* to display λ-isoclines as a function of 2 parameters. Provide juvenile survival rate s_0 as *Variable X*, with *Xmin = 0* and *Xmax = 0.5*. Provide adult fecundity f_2 as *Variable Y*, with *Ymin = 0* and *Ymax = 10*. Click *Exec*. This produces Figure 3.2.

3.7 Environmental stochasticity

Random fluctuations in the environment impact on vital rates, which can be considered as stochastic processes. Stochastic models are studied via Monte Carlo simulation: a large number M of trajectories is cast over some time horizon T. Statistics computed over this set of trajectories give the probabilistic behavior of the population. We give a simple example, using the model `pass_0` (you can load it from the *New from example* dialog).

- Click the *View variable* button . Change the constant expression 0.2 of s_0 into *beta1f(0.2,0.15)*. Type `<return>`. Juvenile survival is now a stochastic variable drawn according to a (variant of) beta distribution, with mean 0.2 and standard deviation 0.15. The beta distribution is used because it takes values in $[0,1]$, which is convenient for survival rates.

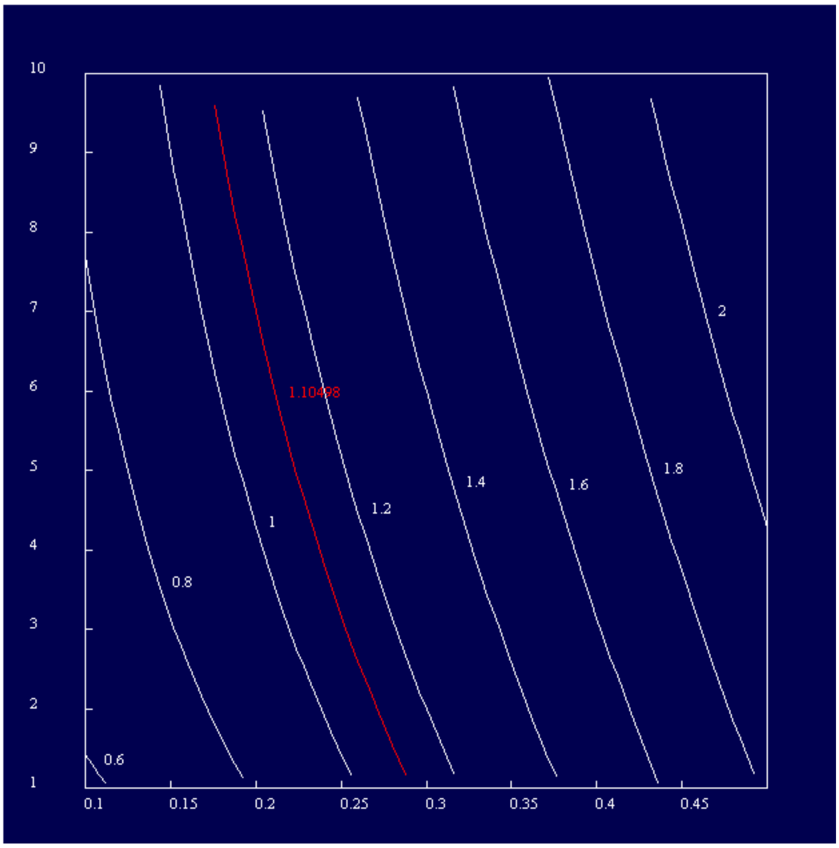


Figure 3.2: Fitness landscape for the passerine model with juvenile survival rate s_0 in X, and adult fecundity f_2 in Y. The red isocline corresponds to the set of (s_0, f_2) values giving the actual growth rate $\lambda = 1.10498$. The steepness of the isoclines reflects the large sensitivity of s_0 as compared to f_2. A small change in s_0 must be compensated by a large change in f_2 to maintain the growth rate.

- Select the *Settings* button in the graphic window. Set the graphics to display s_0 along time t. Click OK. Click the *Run* button ▶. The stochastic trajectory of s_0 appears in the graphic window.
- Change the graphic settings to display population size n along time t. Click the *MonteCarlo* button 🎲 to run the Monte Carlo simulation (50 time steps, 100 trajectories). The mean population size $\bar{n}(t)$ is displayed in the graphic window. Relevant informations appear in the main window (see command **montecarlo** in Section 4).

```
> Montecarlo 50 100
  Mean value [SE]:
    n1 = 491.1314 [332.6684]
    n2 = 176.1816 [84.3060]

  Model passerine (Extinction_threshold = 1.00)
  Non extinct population size (pop*):
```

```
    min = 1.03
    max = 40942.58
    mean = 875.24
    sigma = 4737.1001
    SE = 543.3826

  Probability of escape: 0.0000
  Probability of extinction: 0.2400
  Mean extinction time over extinct trajectories [SE]: 24.3333 [0.8586]

  Stochastic growth rate: 1.000935
  Logarithmic growth rate [SE]: 0.000934 [0.0055]
  Growth rate of the mean pop: 1.072670
  Mean growth rate2 [SE]: 1.023577 [0.006311]
  Growth rate2 of the mean pop: 1.086248

  Mean scaled population structure:
    0.5396    0.4604

  t  pe(t)  pop(t)  SE  pop*(t)  SE
  1  0.0000  23.8  1.1  23.8  1.1
  2  0.0000  25.5  1.7  25.5  1.7
  ( ... )
  48  0.2400  329.8  143.6  433.0  187.3
  49  0.2400  445.2  216.1  584.4  282.5
  50  0.2400  667.3  414.6  875.2  543.4
> Init
```

- $p_e(t)$ = probability of extinction along time
- $pop(t)$ = mean population size (over all trajectories) along time
- $pop^*(t)$ = mean population size over non extinct trajectories along time

The probability of extinction at time t, $p_e(t)$, is computed as the ratio of the number of trajectories that have gone extinct by time t, to the total number M of trajectories. The standard error of variable x is $SE = \frac{\sigma(x)}{\sqrt{M}}$, with $\sigma(x)$ the standard deviation of x and M the number of trajectories in the Monte Carlo.

In the stochastic model, the juvenile survival rate s_0 has the same mean value as in the constant model. However, because of random fluctuations in s_0, the growth rate of the population has decreased from $\lambda = 1.10$ to $\alpha = 1.0$ (stochastic growth rate). The growth rate of the mean population, $\beta = 1.07$, is not a relevant estimator. The inequality $\alpha < \beta$ is (almost) always verified.

3.8 Demographic stochasticity

Demographic stochasticity comes from the chance realization of life cycle transitions by individuals. It is inherent to the demographic process, but its effects are more important when population size is small. Demographic stochasticity is modeled by building a branching process on the matrix relations, using integer-valued distributions. The modeling of demographic stochasticity gives an individual-based feature to the simulation. Individuals are not distinguished by their demographic parameters, that keep their average values, but the fate of individuals is taken into account, via the chance realization of these average parameters.

We use the example model `pass_2s` as example. The underlying linear model corresponds to the `pass_0` example, and to the 2-sex example model `pass_02`. All life cycle transitions are subjected to demographic stochasticity. For life cycle transitions with a result of 0/1 (like survival, sex ratio) the resulting number of individuals is computed using binomial distributions, $n' = \mathrm{binomf}(n, s)$. For reproductive transitions, the number of offspring is computed by summing poisson samples, $n' = \mathrm{poissonf}(n, f)$. Some care is taken to compute the number of matings. While model `pass_0` was of matrix-type, models involving demographic stochasticity such as `pass_2s` must be put in relation-type form. As in the case of environmental stochasticity, the system is studied via Monte Carlo simulation.

- Load `pass_2s` using the *New from example* dialog 🗎.
- Click the *Compile* button 💡. Click the *Settings* button ✥ in the graphic window to parameterize the graphics. Replace $nm1$ by n, remove $nm2$, $nf1$, $nf2$. Replace the actual bounds in Y by [0,400]. Select option *Fix Yscale*. In the *General panel* select *Superimpose*. In the MonteCarlo panel, select *2 sigma*. Click OK.
- Click the *Run* button ▶. The model is run for 50 time steps. The population size along time is displayed in the graphic window.
- Select option *Run* | *Settings*. Change *Random generator seed* from 1 to 2. The system is initialized (**init**). Click the *Run* button ▶. The model is run for 50 time steps, with a different realization of the stochastic process, superimposed to the previous one. Change *Random generator seed* to 3. Click the Run button ▶.
- Select option *Montecarlo* | *Settings* (this is identical to *Run* | *Settings*). Change *Random generator seed from 3 to 1* (the default). Set *Number of trajectories* to 1000 (instead of 100). Click OK. Click 🎲 to run the Monte Carlo simulation (50 time steps, 1000 trajectories).

In the graphic window, the average trajectory with 2σ confidence intervals is superimposed to the previous realizations, which fall within the confidence intervals. The probability of extinction along time appears in the main window.

- Click ▦ to open a text window. Click the *Settings* button in the text window. Replace $nm1$ by nm, replace $nm2$ by nf, remove $nf1$, $nf2$. Set *Sampling interval for text* to 1 (instead of 10). Click OK. Click 🎲 to run the Monte Carlo simulation (50 time steps, 1000 trajectories).

The mean values of the number of males and females along time are displayed in the *Text* window, over all trajectories, and over non extinct trajectories, with standard errors (SE).

3.9 Density dependence

Since resources are limited, the amount available to each individual shrinks as population size increases, leading to density dependence. In the modeling of density dependence, vital rates are regulated by the amount of resources (the carrying capacity K), and by the number of individuals in classes, according to some functional form.

In the following example with 2 age classes, example model `regis`, demographic parameters are regulated using a Ricker type function:

$$x_d = x\exp[-(a_1 n_1 + a_2 n_2)]$$

with x the value of the demographic parameter in absence of density, x_d the regulated value. The contribution of each class to density dependence is expressed by coefficients α_i, with α_i proportional to $1/K$.

- Load `regis` using the *New from example* dialog 🗎.
- Click the Compile button 💡.
- Click button ▶ to run the model (50 time steps by default). Population trajectories are displayed in the graphic window #1: $n1$ and $n2$ along time t.
- Click button ▦ to open graphic window #2. Select window #2, and move it so that window #1 can be seen. Click the *Settings* button ⌘ in window #2 to parameterize the graphics. Change t to $n1$, $n1$ to $n2$, and remove $n2$. Select *line off* in the *General* panel. Click OK.
- Click button · in the main window to initialize the system (t = 0). Select option *Run* | *Settings*. Change *Number of time steps* to 10000, change *Dt text interp* to 1000. Click OK. Click ▶ to run the model for 10000 time steps, with output in the main window every 1000 times steps. A strange attractor appears in window #2 (Figure 3.3),with the corresponding population trajectories in window #1.
- Click button ▤ to view the variables. Change the *expression* of variable r to 50. Type `<return>`. The system is initialized (**init**). Click the *Run* button ▶. A single point equilibrium is displayed (cover figure).
- Change r to 60. Type `<return>`. Click the Run button ▶. A quasi-cycle is displayed.
- Select the *Tools* | *Spectrum* option. Change *Variable* to n. Click the *Run* button in the *Spectrum* window. The power spectrum of population size is displayed, showing a discrete set of frequencies. Change the expression of r to 115. Click the *Run* button in the Spectrum window. A set of continuous frequencies appears.
- Select the *Tools* | *Lyapunov* option. Click the *Run* button in the Lyapunov window. Estimators of the Lyapunov exponent r are given, showing the presence of weak chaos ($r > 0, r \sim 0$).

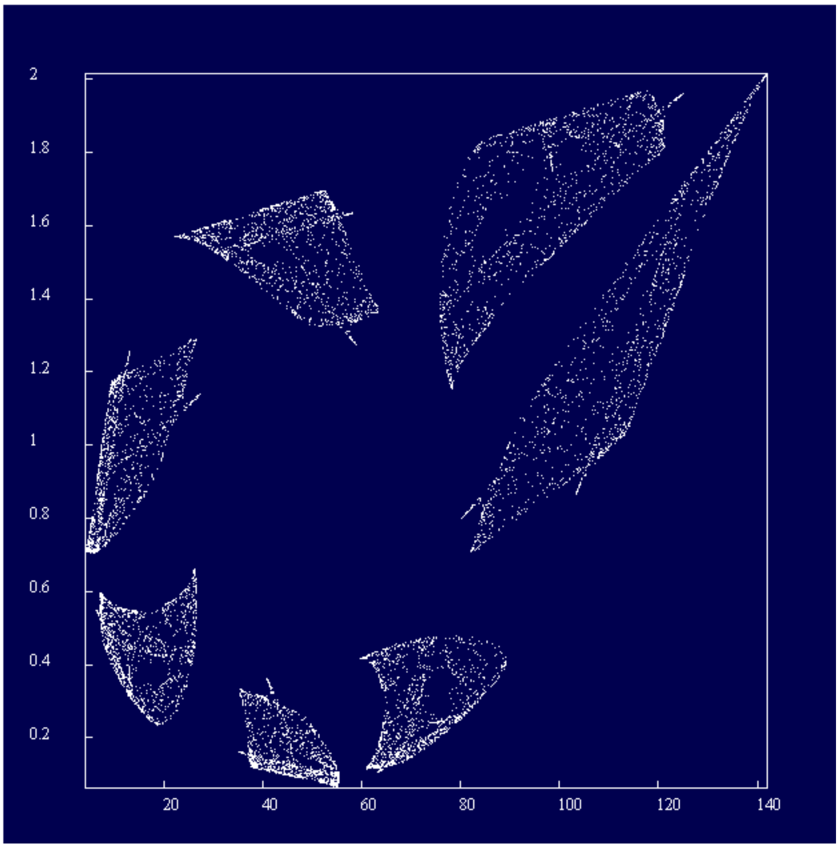


Figure 3.3: Density dependent population dynamics produce this strange attractor (model from [Cazelles and Ferrière, 1992]).

4 OBJECTS

ULM models are built from objects related by mathematical functions, and processed along time by the ULM kernel. The models are described in an input text file (`*.ulm` file), using a declaration language. The model file is processed by the *Compile* command (💡), and searched for syntax errors. When the syntax is correct, the model can be run (▶).

There are 6 types of objects handled by the ULM kernel with corresponding keywords:

defmod	*declaration of model*
defmat	*declaration of matrix*
defvec	*declaration of vector*
defrel	*declaration of relation*
defvar	*declaration of variable*
deffun	*declaration of function*

Each object is referenced by a user chosen name (names begin with any letter 'a' to 'z'). Other keywords specify mathematical operators or functions (see Section 6).

- Declarations of objects must be separated by blank lines.
- Lines beginning with '{' are comment lines and are not processed.
- The model file must begin with the declaration of a model (**defmod**).
- The declaration of a model must precede the declarations of its associated object, matrix and vector, or relations.
- Relations may be declared without any link to a model.
- All variables and functions must be declared explicitly.
- Letters are converted to low case; the interpreter is not case sensitive.

defmod declaration of model

A model describes a population whose dynamics are driven by a set of discrete time relations. These relations can be put in matrix form, using a population matrix and a population vector.

matrix-type model

defmod model_name(k)	*declaration of model of size k*
mat: aaa	*name of matrix*
vec: vvv	*name of vector*

Example: file `pass_0.ulm`.

defmod passerine(2)	*model passerine of size 2*
mat: a	*matrix a*
vec: w	*vector w*

The matrix and vector are to be declared elsewhere in the file, using the **defmat** and **defvec** keywords.

relation-type model

defmod model_name(k)	*declaration of model of size k*
rel: rel1, ..., relk	*names of k relations*

Example: file `met_esd.ulm`.

defmod metepeira_esd(5)	*declaration of model of size 5*
rel: r1, r2, r3, r4, r5	*5 relations: r1, ..., r5*

The relations are to be declared elsewhere in the file, using the **defrel** keyword.

A single model file may include several models.

defmat declaration of matrix

defmat matrix_name(k)	*declaration of matrix of size k*
a11, . . . , a1k	*first line of matrix entries*
a21, . . . , a2k	*second line of matrix entries*
⋮	
ak1, . . . , akk	*k-th line of matrix entries*

Example:

defmat a(2)	2 × 2 *matrix for passerine model file* `pass_0.ulm`
sigma*s0*f1, sigma*s0*f2	*matrix entries*
s, v	

defvec declaration of vector

defvec vector_name(k)	*declaration of vector of size k*
n1, . . . , nk	*names of k variables that are the vector entries*

Example:

defvec w(2)	*population vector for passerine model file* `pass_0.ulm`
n1, n2	*names of vector variables*

defrel declaration of relation

defrel relation_name	
var_name = expression	*expression for the relation*

Example: file `pass_2s.ulm`

defrel rm1
nm1 = **binomf**(pf1m+pf2m, sm0)

defrel rm2
nm2 = **binomf**(nm1, sm) + **binomf**(nm2, vm)

In this example, variables $nm1$ and $nm2$ are relation-variables. From one time step to the next, relation-variables are updated in parallel (and not sequentially), as would be the case in matrix form.

defvar declaration of variable

defvar variable_name = expression

There is one and only one predefined variable, whose name is t for ‘time’. Variable t takes values 0, 1, 2, ... as the system is run. Other variables are declared by the user.

If *variable_name* is the name of a variable pertaining to a vector (vector-variable, **defvec**) or to a relation (relation-variable, **defrel**), then *expression* must be a real number, which is the initial value of the variable.

Examples:

defvar s0 = 0.2   *constant*

defvar n1 = 100   *relation-variable with initial value 100*

defvar phi = (1+**sqrt**(5))/2   *constant*

defvar x = **gaussf**(2, 0.1)   *random variable*
normal distribution with mean 2 and standard deviation 0.1

defvar w = **if**(t > 10, x, 0)   *conditional*

defvar n1 n2 = 100   *shared declaration*

deffun declaration of function

deffun function_name(arg1, ..., argN) = expression

The arguments of the function have the names $arg1, \ldots, argN$.

Examples:

deffun som(v, n) = (1 - v^(n+1)) / (1 - v)   *sum of a geometric series*

deffun fac(n) = **if**(n, n*fac(n-1), 1)   *recursive definition of the factorial*

deffun alpha(s1, s2) = c*(1 - 1/(1 + d***exp**(-k*(s1 - s2))))

4.1 The update procedure, from one time step to the next

At each time step of the ULM simulation, population matrices, population vectors, relations, and variables are updated in a specific order given below.

- The right hand side expressions of all relations are computed.
    - For matrix-type models, the entries of the associated matrices are computed, then the product with the population vector.
    - For relation-type models, the expressions of the associated relations are computed.
    - Expressions of relations that are not associated with a model are computed.
- Relation-variables and vector-variables are updated.
    - Vector-variables associated with matrix-type models are updated.
    - Relation-variables associated with model relations are updated.
    - Relation-variables associated with other relations are updated.
- Time t is updated ($t = t + 1$). All remaining variables are updated according to their dependencies.

At initialization (**init**), the ULM program builds a hierarchy of all variables, according to their dependencies. It is checked whether there are circular definitions of variables. If this is the case, the message “cycling definition of variable xxx” warns the user that the computations are not reliable. From the hierarchy an order of computation over all variables is established. This order is used throughout the simulation to update the variables consistently (use button to see the order of evaluation).

5 COMMANDS

Once the ULM model file is compiled (using 💡), commands allow to study the model interactively (run simulations, set graphics, . . .). Most commands rely on clicking the appropriate buttons (like ▶ to run the model, ▪ to initialize the model), and can be parameterized using the appropriate *Settings* options (like *Run* | *Settings*). Most commands can also be entered in the small *Interpreter* panel of the main window. The syntax is:

command_name $p1\ p2\ldots$

where **command_name** is the name of the command, and $p1, p2, \ldots$ are parameters of the command. For example, after typing:

run 100 10

the system is run for 100 time steps with output every 10 time steps in the large panel of the main window. Trajectories are displayed in the graphic windows. Equivalently, select the *Run* | *Settings* option, set *Number of time steps* to 100, and *Dt text interp* to 10. Then click the *Run* button ▶.

Parameters of commands are names, integer or real values, or may be empty. Each command can be abbreviated by a single character. For example,

graph t n1 n2 is equivalent to **g** t n1 n2

and sets the trajectories to be displayed in graphic window #1, in this case the values of variables $n1$ and $n2$ along time t.

Graphics can also be parameterized using the *Settings* option ✕ in the graphic windows.

In this section, the list of commands is sorted in alphabetical order. The mention 'on/off' means that the command works in an *on/off* manner. For example, typing:

addgraph

allows to superimpose graphs in graphic window #1 ('Addgraph ON'), and typing again

addgraph

disables this option ('Addgraph OFF').

The mention 'graph' indicates that the command is related with graphics. Optional command parameters are between < >.

5.1 Command file

ULM simulations can be performed in absence of the user using a command file, a text file containing the commands you would have typed in the *Interpreter* panel.

Example 1 Command file associated with the example model `regis`

graph n1 n2	*set graphics, phase portrait (n1, n2)*
line	*set line OFF*
run 10000 1000	*run 10000 time steps, with results every 1000 time steps*
savegraph regis.bmp	*save graphics (strange attractor) in bitmap file regis.bmp*

Example 2 Command file associated with the example model `pass_2s`

graph t n	*set graphics*
text t n	*set output of the main window*
change nm1 2	*set population size and structure*
change nm2 2	
change nf1 2	
change nf1 2	
change nf2 2	
montecarlo 100 10000	*run Monte Carlo simulation*
change nm1 4	*set population size and structure*
change nm2 4	
change nf1 4	
change nf2 4	
montecarlo 100 10000	*run Monte Carlo simulation*

File names are passed as parameters of the ULM program (`ulm.exe` under Windows, `ulm` under Linux or macOS) with the following syntax:

```
ulm.exe  model_file  command_file  output_file
```

Paths to parameter files refer to the directory where the ULM program is installed. For example,

```
ulm.exe  models/regis.ulm  models/regis.in  regis.out
```

The output text file is optional: if provided, results displayed in the main window will be stored in it.

Several simulations with different model files can be grouped using a batch file (`.bat` file under Windows, `.sh` shell script under Linux). Simply create a text file containing the relevant commands for executing the ULM program.

5.1.1 Windows

- Drag and drop the `ulm.exe` icon to the desktop, creating a shortcut to `ulm.exe`.
- Right click the shortcut to access to its properties, and add to the name of the program (`ulm.exe`) the names of the files, for example,

  ```
  C:/ulm/ulm.exe  models/regis.ulm  models/regis.in  regis.out
  ```

- Double-click the shortcut to run the ULM simulation.

Example 3a Batch file associated with examples 1 and 2:

```
ulm.exe  models/regis.ulm    models/regis.in  regis.out
ulm.exe  models/pass_2s.ulm  pass_2s.in       pass_2s.out
```

The file name must have the `.bat` extension, for example `MySimul.bat`. In this example, the batch file is located in the same directory as `ulm.exe` and `pass_2s.in`. The files `regis.ulm`, `regis.in` and `pass_2s.ulm` are in the subdirectory `models`. Double-click the batch file `MySimul.bat` to execute the simulations.

5.1.2 Linux

Open a terminal (shell) window, and type a command line specifying the ULM program parameters. For example, assuming that you are in the directory where the program `ulm` is located:

```
./ulm  ./models/regis.ulm  ./models/regis.in  regis.out
```

Example 3b Shell script associated with examples 1 and 2:

```
./ulm  ./models/regis.ulm    ./models/regis.in  regis.out
./ulm  ./models/pass_2s.ulm  pass_2s.in         pass_2s.out
```

The file name must have the `.sh` extension, for example `MySimul.sh`. In this example, the script is located in the same directory as `ulm` and `pass_2s.in`. The files `regis.ulm`, `regis.in` and `pass_2s.ulm` are in the subdirectory `models`. Type

```
sh  MySimul.sh
```

to execute the simulations.

Addgraph on/off graph

abbreviation:	+
other name:	**add**
syntax:	**addgraph**
function:	Superimpose graphics
default:	*off*

- Use option *Superimpose* in the graphic windows *Settings*.
- Graphic window #i can be selected using the **window** command.

note When addgraph is *on*, do not resize the graphic window.

Example file `met_esd.ulm`

?	**change** nmax 200	*change value of population ceiling*
?	**graph** t n	*plot population size n as a function of time* **t**
?	**run** 100	*run 100 time steps*
?	**yscale**	*fix bounds in Y axis (current bounds)*
?	**addgraph**	*superimpose graphics*
	Addgraph ON	
?	**change** nmax 50	*back to initial value of population ceiling*
	Init	
?	**run** 100	*run 100 time steps* *Appreciate how the trajectory separates from* *the previous one once the population ceiling is reached*

Border on/off graph

abbreviation:	b
syntax:	**border**
function:	Graphic scales are drawn if *on*
default:	*on*

- Use option *Hide* in the graphic windows *Settings*.
- Graphic window #i can be selected using the **window** command.

Changevar

abbreviation: c
other name: change
syntax: **change** *var expr*
parameters: *var* variable name
expr mathematical expression
function: Replace actual expression of variable *var* by new expression *expr*. System is initialized (**init**).

- Use button to display the model variables. The expression of any variable can be changed by clicking in the corresponding field.

note Vector-variables and relation-variables must be set to a real number, which is their initial value.

Example 1: file `pass_2s.ulm`

?	**graph** t n	*plot population size n as a function of time t*
?	**montecarlo** 50 1000	*run Monte Carlo simulation*
		probability of extinction pe = 0.616
?	**change** s0 beta1f(0.2, 0.15)	*make juvenile survival rate s0 stochastic*
?	**montecarlo** 50 1000	*run Monte Carlo simulation*
		probability of extinction pe = 0.911

Confidence interval

function: For matrix models, compute standard deviation σ_λ of the growth rate λ, given the standard deviations σ_{ij} on the non-zero matrix entries a_{ij}, assumed to vary independently. The formula, involving the sensitivities, is

$$\sigma_\lambda^2 = \sum_{i,j} \left(\frac{\partial \lambda}{\partial a_{ij}} \sigma_{ij} \right)^2 .$$

Assuming that λ follows a normal distribution, this provides the confidence interval $\lambda \pm z\sigma_\lambda$, where z is chosen by the user.

By default, $z = 1$, giving a 68 % confidence interval. Entering $z = 1.96$ gives a 95 % confidence interval.

- Select the *Matrix | Confidence interval* option to get the *Confidence interval* window, in which the standard deviations σ_{ij} on the matrix entries can be entered (the σ_{ij}'s are set to 0.1 by default).

note This option is not accessible from the command line.

Confidence interval 2

function: For matrix models, compute the standard deviation σ_λ of the growth rate λ, given standard deviations σ_{x_i} on demographic parameters $x_1, \ldots, x_p$, assumed to vary independently. The formula is

$$\sigma_\lambda^2 = \sum_{i=1}^{p} \left(\frac{\partial \lambda}{\partial x_i} \sigma_{x_i} \right)^2 .$$

Assuming that λ follows a normal distribution, this provides the confidence interval $\lambda \pm z\sigma_\lambda$, where z is chosen by the user.

By default, $z = 1$, giving a 68 % confidence interval. Entering $z = 1.96$ gives a 95 % confidence interval.

- Select the *Matrix | Confidence interval 2* option to get the *Confidence interval 2* window, in which the standard deviations σ_{x_i} of the user defined demographic parameters can be entered.

note This option is not accessible from the command line.

Correlation graph

abbreviation: **o**
other name: **correl**
syntax: **correl** $< var1 > < var2 >$
function: Display cross-correlation of variables $var1$ and $var2$,
or autocorrelation of variable $var1$, if $var2$ is not provided.
The system is run for 400 time steps,
100 values of correlation are displayed.

- Select the *Tools* | *Correlation* option to get the *correlation* window, in which the number of time steps and the number of values can be parameterized.

note $var1$ and $var2$ should correspond to stationary processes.

Distribution on/off graph

abbreviation:	**u**
other name:	**distrib**
syntax:	**distrib** < $delta$ >
parameter:	$delta$ real number > 0
default:	$delta = 1$
function:	Display distributions of variables specified by the **graph** command. For variable x: number of values of x such that $delta * j \leqslant x < delta * (j+1)$ for $j = 1, \ldots, 10000$ For the **run** command, distribution along time. For the **montecarlo** command, distribution across trajectories at time horizon.
default:	*off*

- Use alternatively option *distrib* in graphic windows *Settings* .
- Select option *include 0* to include the value 0 in the distribution.
- Graphic window #i can be selected using the **window** command.

Example 1 file `pass_2s.ulm`

```
? graph t n              note: t is dummy for distrib
? distrib 100            set distribution mode with delta = 100
  Distribution mode ON
? run 100                display distribution of n(t) along time
? montecarlo 100 1000    display distribution of n(t) at time t = 100,
                         over 1000 trajectories
```

Example 2 file `regis.ulm`

```
? graph t n              note: t is dummy for distrib
? distrib 0.1            set distribution mode with delta = 0.1
  Distribution mode ON
? change r 50            lead to point equilibrium
? run 1000 1000          display distribution of n over 1000 time steps
? change r 60            lead to quasi-circle
? run 10000              display distribution of n over 10000 time steps
? change r 110           lead to chaos
? run 10000              display distribution of n over 10000 time steps
```

Erase ▪ graph

abbreviation:	**e**
other name:	**clear**
syntax:	**erase**
function:	Clear graphics (window #1 or graphic window selected by command **window**).

- Use alternatively button ▪ in graphic windows.

File on/off

abbreviation:	**f**
syntax:	**file** *file_name* $x1 \ldots xN$
parameters:	*file_name* name of file $x1,\ldots,xN$ names of variables
function:	Create text file *file_name* and store values of variables in the file as the model is run (**run** or **montecarlo** command). When the variable names are not given, the file *file_name* is closed. Storage differ according to the **run** or **montecarlo** command.

For the run command The format of each line in the file is:

```
t v1 v2 ...vN
```

where $v1,\ldots,vN$ are the values of variables $x1,\ldots,xN$ at time t. There is a new line in the file at each time step that is a multiple of the second parameter Δ of the **run** command (see **run** command). For example, with the command

? **run** 1000 10

values are stored every Δ = 10 time steps.

For the montecarlo command **montecarlo** T M, with T the number of time steps and M the number of trajectories. The format of each line in the file is:

```
j v1 v2 ...vN
```

where $v1,\ldots,vN$ are the values of variables $x1,\ldots,xN$ at time T in the j-th trajectory ($j = 1,\ldots,M$). There is a new line in the file for each trajectory.

- Up to 5 files can be created in a session.
- Up to 10 variables can be stored simultaneously in a file.
- The path of the file can be specified (the default path is where the ULM program is located, usually `c:\ulm`). Example:

  ? **file** c:\myfolder\myfile.txt x1 x2.

  The number of digits after the decimal point can be specified using the separator ':' (the default precision is 4). For example, after the command

```
? file myfile.txt x1:10 x2
```

variable $x1$ will be stored with 10 digits after the decimal point, and variable $x2$ with 4 digits after the decimal point.

Example 1: model file `regis.ulm`

```
  ?   newvar lamb lambdaf(1,1)   create new variable lamb whose value is the
                                 dominant eigenvalue of the model
  ?   file regis.txt lamb:6      open file regis.txt to store variable lamb

                                 (precision = 6)
      File regis.txt opened
  ?   run 100 1                  run model (100 time steps, Δ = 1)
      (...)                      values are stored in the file every time step (Δ = 1)
  ?   file regis.txt             close file regis.txt
      File regis.txt closed
```

File `regis.txt` looks:

```
 0     4.902861
 1     4.902861
 2     0.183002
 3     1.897385
 4     3.756156
 5     0.103296
 6     0.969797
(...)
```

Example 2: model file `allee.ulm`

```
  ?   file allee.txt p0 p1      open file allee.txt to store variables p0 and p1
      File allee.txt opened
  ?   montecarlo 100 100        run model (100 time steps, 100 trajectories)
       (...)                    values are stored in the file for each trajectory
  ?   file allee.txt            close file allee.txt
      File allee.txt closed
```

File `allee.txt` looks:

```
 1     0.9800   0.8600
 2     0.7800   0.7800
 3     0.9200   0.9200
 4     0.9000   0.7400
(...)
```

Graph ▦ graph

abbreviation:	**g**
syntax:	**graph** x $y1 \ldots yN$
parameters:	x, $y1, \ldots, yN$ names of variables
function:	Display variables $y1, \ldots, yN$ as a function of variable x in graphic window. If distribution mode is *on*, distributions of $y1, \ldots, yN$ are displayed.

See also **addgraph, border, distribution, erase, line, savegraph, window, xscale, yscale**

- Up to 6 graphic windows, numbered #1 to #6, can be created using button ▦.
- Each graphic window can be parameterized using the *Settings* option ✕.
- The **graph** command operates on window #1 unless window #i has been selected using the **window** command. It is useful in command files.

Note When the number of time steps is larger that 10000, a sampling of the trajectories is performed (see Dt in the graphic window status bar). For example, for 100000 time steps a point is taken every $Dt = 10$ time steps.

Example 1: file `regis.ulm`

?	**graph** n1 n2	*set graphics for phase portrait*
?	**change** r 110	*change bifurcation parameter*
?	**run** 10000 1000	*display strange attractor*
?	**change** r 60	*change bifurcation parameter*
?	**run** 10000	*display limit cycle*

Example 2: file `met_esd.ulm`

- Click button ✕ in graphic window. Change $n1$ to ntt, change $n2$ to $nmax$, remove $n3$, $n4$. Click OK.
- In the small *Interp* panel of the main window, type 'run 100'. Appreciate how the trajectory of ntt is bounced when the population ceiling $nmax$ is overshot.
- Type 'init 2'. Click the Run button ▶ to display another realization of the process.

Help

abbreviation:	**h** or **?**
syntax:	**help** < *xxx* >
parameters:	*xxx* name of command or mathematical function
function:	Give succinct on line information about commands and mathematical functions **help**: list of all commands and mathematical functions. **help** *xxx*: short description of command *xxx* or mathematical function *xxx*.

Example

? **help** lambdaf

```
lambdaf(i,j)
modulus of jth eigenvalue of ith model
(in the order of declaration)
domain: 1 <= i <= model_nb
1 <= j <= size of ith model
lambdaf(i,1) = dominant eigenvalue of ith model
```

Init

abbreviation:	**i**
syntax:	**init** <*j*>
parameters:	*j* integer ≥ 0, random generator seed
function:	**init**: initialize $t = 0$, variables are reset to their initial values, random generator is reset to its initial value (called seed). **init** *j*: init + seed initialized to j, corresponding to the j-th trajectory of the Monte Carlo procedure. **init** *1*: init + back to the default seed ($j = 1$).

Note **init** is performed automatically after the following commands: **changevar**, **montecarlo**, **newvar**.

- Command init can be performed using button in the main window.
- The random generator seed can be set using the *Run* | *Settings* option.

Example file `pass_2s.ulm`

```
?  graph t n
?  init
   Init
?  montecarlo 50 100            give probability of extinction estimate
   (...)                        pe = 0.68 (at time 50)
?  init 500
   random generator seed -> 500
   Init
?  montecarlo 50 100            another estimation
   (...)                        pe = 0.60
?  init 1
   random generator seed -> 1
   Init
?  montecarlo 50 1000           back to first simulation, better estimate
   (...)                        pe = 0.616
?  init 500
   random generator seed -> 500
   Init
?  montecarlo 50 1000           back to 2nd simulation, better estimate
   (...)                        pe = 0.615
```

Line on/off graph

abbreviation: **l**
syntax: **line** <*col*>
parameters: *col* integer in [1,...,16], line color
function: If *on* lines of color col are drawn
between consecutive points in graphic window #1
(or window #i specified by the **window** command).
Useful in command files.
default: *on*

- Use alternatively option *line off* in graphic windows *Settings* .
- Use color specification in graphic windows *Settings* , by clicking on the colored button next to the graphic variables panels Y1 Y2 Y3 Y4.

Example file `pass_2s.ulm`

? **graph** t n
? **line** 1
? **yscale** 0 400
? **run** 50 *display red trajectory*
? **addgraph**
? **line** 2
? **init** 2
? **run** 50 *superimpose green trajectory*

Lyapunov

abbreviation:	**q**
syntax:	**lyap** < *model* >
parameters:	*model* name of a model (default first model)
function:	Compute an estimator of the first Lyapunov exponent r of model *model*. The system is run for 1000 time steps, with output every 100 time step. $r < 0 \Leftrightarrow$ fixed point equilibrium or cycle $r \sim 0 \Leftrightarrow$ quasi-cycle $r > 0 \Leftrightarrow$ chaos

Note The command is intended for deterministic regulated systems. For a constant matrix, $r = \ln(\lambda)$, λ the dominant eigenvalue of the matrix.

- Select the *Tools* | *Lyapunov* option to get the Lyapunov exponent window, in which the number of time steps and the time lag for output can be parameterized.

Example file `regis.ulm`

?	**change** r 50	
?	**lyap**	*estimator of the lyapunov exponent (*$r < 0$*, point equilibrium)*
?	**change** r 60	
?	**lyap**	*estimator of the lyapunov exponent (*$r \sim 0$*, quasi-cycle)*
?	**change** r 110	
?	**lyap**	*estimator of the lyapunov exponent (*$r > 0, r \sim 0$*, weak chaos)*

Montecarlo

abbreviation:	**m**
other name:	**monte**
syntax:	**montecarlo** *T M* <*Ext*> <*Esc*>
parameters:	T integer > 0, number of time steps M integer > 0, number of trajectories Ext real number > 0, extinction threshold (default $Ext = 1$) Esc real number > 0, escape threshold (default $Esc = 10^7$)
function:	Monte Carlo simulation. M trajectories are run over a time horizon of T time steps. System is initialized at the end (**init**).

- Monte Carlo simulation is parameterized using the *MonteCarlo* | *Settings* option.
- Monte Carlo graphics are parameterized using the *Settings* option in the graphic windows.
- Monte Carlo outputs are parameterized using the option *Settings* in the text windows.
- Press Ctrl-Alt simultaneously to break simulation (with the main window selected).

Notes

- Mean trajectories over M trajectories are displayed in the graphic windows (with min, max and $\pm 2\sigma$ intervals if requested).
- Mean values along time with standard errors are displayed in the text windows (including or excluding extinct trajectories).
- If distribution mode is *on*, distributions of trajectories at time T are displayed.
- j-th trajectory whose population size $n_j(t) < Ext$ is declared extinct (at time t), but computed to the end (default $n_j(t) < 1$).
- j-th trajectory whose population size $n_j(t) > Esc$ is declared escaped (at time t), but computed to the end.
- Population size is computed as the sum of the values of the vector-variables (or the relation-variables) of the model.
- For each model: probability of extinction along time, mean time to extinction (computed over extinct trajectories), probability of escape, mean escape time (computed over escaped trajectories), growth rates, non extinct population size values, mean population structure.

- Stochastic growth rate = $\exp(a)$ where a is the average of the logarithmic growth rates of M trajectories, computed as

  $$a = \frac{1}{M}\sum_{j=1}^{M}\left[\frac{ln(n_j(T)) - ln(n_j(0))}{T}\right].$$

  Relevant estimator for pure environmental stochasticity.

- Mean growth rate = average of the growth rates of M trajectories, computed as

  $$\frac{1}{M}\sum_{j=1}^{M}\exp\left[\frac{ln(n_j(T)) - ln(n_j(0))}{T}\right].$$

- Growth rate of the mean pop = growth rate of the average trajectory, computed as

  $$\exp\left[\frac{ln(\bar{n}_j(T)) - ln(\bar{n}_j(0))}{T}\right] \quad \text{with} \quad \bar{n}(t) = \frac{1}{M}\sum_{j=1}^{M} n_j(t),$$

  $\bar{n}(t)$ the average trajectory.

- Mean growth rate2 = average of growth rates of *non extinct* trajectories, computed as

  $$\frac{1}{M^*}\sum_{j=1}^{M^*}\frac{n_j^*(1) + \cdots + n_j^*(T)}{n_j^*(0) + \cdots + n_j^*(T-1)},$$

  where $n_j^*(t)$ is a non extinct trajectory.

- Growth rate2 of the mean pop = growth rate of average *non extinct* trajectory, computed as

  $$\frac{\bar{n}_j^*(1) + \cdots + \bar{n}_j^*(T)}{\bar{n}_j^*(0) + \cdots + \bar{n}_j^*(T-1)} \quad \text{with} \quad \bar{n}^*(t) = \frac{1}{M^*}\sum_{j=1}^{M^*} n_j^*(t),$$

  $\bar{n}^*(t)$ the average non extinct trajectory. Relevant estimator for pure demographic stochasticity.

Example 1 file `pass_2s.ulm`

```
?  graph t n
?  text t n
?  montecarlo 50 1000    run Monte Carlo simulation
                         50 time steps, 1000 trajectories
 (...)
growth rate2 of the mean pop = 1.0254
growth rate estimator for demographic stochasticity
```

```
t    pe(t)    pop(t)  SE   pop*(t)  SE
10   0.0020   47.1    0.8  47.2     0.8
20   0.1410   42.1    1.3  49.0     1.4
30   0.3690   42.5    2.0  67.4     2.7
40   0.5040   48.8    3.0  98.4     5.2
50   0.6160   63.9    4.8  166.3    10.7
```

pe = probability of extinction, pop = mean pop size, SE = standard error, pop* = mean pop size over non extinct trajectories

? **view** cc — *coefficient of reduction in the number of matings*
cc = 0.95
? **change** cc 1 — *no reduction in number of matings*
? **montecarlo** 50 1000 — *run Monte Carlo simulation*
50 time steps, 1000 trajectories
`(...)`
growth rate2 of the mean pop = 1.0812
probability of extinction at time 50 = 0.084
(much lower than 0.616)

Example 2 file `pass_2s.ulm`

Initial population size is 48 individuals

? **graph** t n
? **text** t n
? **montecarlo** 50 1000 30 — *run Monte Carlo simulation*
50 time steps, 1000 trajectories
extinction threshold = 30
`(...)`
probability of extinction = 0.760
Probability to get less than 30 individual by time 50
mean population size at time 50 [SE] = 64 [5]
mean population size at time 50 over non extinct trajectories [SE] = 247 [15]
? **change** nm1 24 — *change initial population size to 96 individuals*
? **change** nm2 24
? **change** nf1 24
? **change** nf2 24
? **montecarlo** 50 1000 30

```
probability of extinction = 0.121
mean population size at time 50 [SE] = 564 [16]
mean population size at time 50 over non extinct trajectories [SE] = 640 [17]
```

Probability to get less than 30 individual by time 50

Example 3 file `pass_2s.ulm`

Initial population size is 48 individuals

? **graph** t n

? **montecarlo** 50 1000 1 100 *run Monte Carlo simulation*
50 time steps, 1000 trajectories
extinction threshold = 1
escape threshold = 100

(...)

```
probability of escape = 0.212
mean escape time = 21
probability of extinction = 0.616
mean extinction time = 29
```

Respectively, probability to get more than 100 individuals by time 50, and probability to get less than 1 individual by time 50.

Example 4 file `met_esd.ulm`

- Click button ⤧ in graphic window. Change $n1$ to n, remove $n2$, $n3$, $n4$. Select options $MinMax$ and 2 $sigma$. Click OK.
- Click button 🎲 to run Monte Carlo simulation (**montecarlo** 50 100 by default). Mean trajectories appear with 2σ confidence intervals, maximum and minimum values.
- Select option *Montecarlo | Settings*. Change *Number of trajectories* to 1000. Click OK. Click button 🎲 to run Monte Carlo simulation (now **montecarlo** 100 1000).

Newvar

abbreviation:	**n**
other name:	**new**
syntax:	**newvar** var $expr$
parameters:	var name of a variable $expr$ mathematical expression
function:	Creation of a new variable with name var and expression $expr$. System is initialized (**init**).

Example 1 file `pass_0.ulm`

?	**newvar** p1 n1/n	*create variable* $p1$ *= proportion in age class 1*
?	**newvar** p2 n2/n	*create variable* $p2$ *= proportion in age class 2*
?	**graph** t p1 p2	
?	**run** 20	*compare with stable age distribution (command* ***property****)*

Example 2 file `pass_0.ulm`

?	**newvar** g lambdaf(1,1)	*create variable g* *whose value is the dominant eigenvalue*
?	**xscale** 0 1	*set bounds in X*
?	**yscale** 0 1	*set bounds in Y*
?	**addgraph**	*superimpose graphics*
?	**graph** sigma g	*plot g as a function of* $sigma$
?	**parameter** sigma 0 1 0.1	*make primary sex ratio* $sigma$ *vary*
?	**skip** 1	*skip one time step*
?	**run** 1	*display growth rate as a function of* $sigma$

Example 3 file `pass_2s.ulm`

?	**newvar** pe n < 1	*create variable pe = if* $n < 1$ *then 1 else 0*
?	**graph** t pe	*set graphics*
?	**yscale** 0 1	*fix bounds in Y*
?	**montecarlo** 100 1000	*display pe = probability of extinction along time* *(as average trajectory)*

Parameter

abbreviation:	**a**
other name:	**param**
syntax:	**param** *var min max step*
parameters:	*var* name of a variable, used as a parameter *min* real number, lower bound of variation *max* real number, upper bound of variation *step* real number > 0, incremental step
function:	Variable *var* will be varied between *min* and *max* by step *step*, when the **run** command is executed. The system will be run for each value of the parameter variable *var*.

Note After the **run** command, parameter is *off.* System is initialized (**init**).

Example 1 file `pass_0.ulm`

```
? xscale 0 100              fix bounds in X
? yscale 0 1000             fix bounds in Y
? addgraph                  superimpose graphs
? graph t n                 plot n along time t
? parameter s0 0 0.5 0.1    declare s0 as parameter
  Parameter ON              min = 0, max = 0.5, step = 0.1
? run 50                    display n(t) for values of the parameter s0
```

Example 2 file `regis.ulm`, bifurcation diagram

```
? xscale 100 110            fix bounds in X
? yscale 0 200              fix bounds in Y
? addgraph                  superimpose graphics
? line                      no lines between points on graphics
? skip 1900                 do not display transients (1900 time steps)
? param r 100 110 0.02      r is the bifurcation parameter,
                            min = 100, max = 110, step = 0.02
? graph r n                 display n as a function of r
? run 2000                  run the system for 2000 time steps,
                            for each value of the parameter r
```

Property P

abviation: **p**
other name: **prop**
syntax: **property** < mat >
parameters: mat name of a matrix (default first matrix)
function: Give properties of the matrix mat

Matrix Properties

- irreducibility, primitivity, type: Leslie, extended Leslie, size-classified, multisite, time dependent, vector dependent (density dependence or frequency dependence), random.
- eigenvalues λ_i, damping ratio ρ, period P.
- left and right eigenvectors associated with the dominant eigenvalue λ: reproductive value V and population structure W).
- other demographic quantities: net reproductive rate R_0, generation times T, T_c, $\bar{T}$, entropy S, entropy rate H.

Note The command does not work for a relation-type model. If the matrix is not constant, properties of the matrix at current time t are displayed.

- Use the *Property* button P to access to the matrix properties.
- The *Matrix* | *Age* option is intended for size-classified matrices, and provides the time spent in stages (see [Barot et al., 2002]). This option is also meaningful for age-classified matrices.
- The *Matrix* | *Multisite* option is intended for block matrices, used in the modeling of metapopulations with migrations between patches (see [Lebreton, 1996]).
- Other *Matrix* options (*Sensitivities, Stochastic sensitivities, Landscape*) are detailed in the command **sensitivity**.

Example file `pass_0.ulm`

?	**property**	*ask for matrix properties*
		dominant eigenvalue $\lambda = 1.1050$
?	**change** s0 beta1f(0.2,0.15)	*make matrix stochastic*
?	**property**	*gives properties of matrix at time* $t = 0$
	(...)	
?	**run** 100	
?	**property**	*gives properties of matrix at time* $t = 100$

Run ▶

abbreviation:	**r**
syntax:	**run** $T < \Delta >$
parameters:	T integer > 0, number of time steps (default $T = 100$) Δ integer > 0, number of steps for text display (default $\Delta = 10$)
function:	Run the models for T time steps with output every Δ time steps.

- The **run** command is parameterized using the *Run* | *Settings* option.
- Graphics are parameterized using the graphic windows *Settings* option.
- Results are parameterized using the text windows *Settings* option.
- Press Ctrl-Alt simultaneously to break simulation (with the main window selected).

Notes

- Trajectories are displayed in graphic windows, numerical values are displayed in text windows.
- Growth rate estimator of the models from time $t = T_0$:

$$\hat{\lambda} = \exp\left[\frac{\ln(n(T+T_0)) - \ln(n(T_0))}{T}\right],$$

with n the number of individuals along time (sum of relation-variables values for a relation-type model, sum of vector entries for a matrix-type model).

Example 1 file `pass_0.ulm`

```
?  property                               dominant eigenvalue λ = 1.104975
?  run 20                                 run 20 time steps
   Model passerine -> pop = 150.2
    growth rate from [t = 0] -> 1.106061
?  run                                    run 20 more time steps
   Model passerine -> pop = 1105.8
    growth rate from [t = 0] -> 1.105518
    growth rate from [t = 20] -> 1.104975
   compare growth rate estimator and λ
```

Example 2 file `regis.ulm`

?	**graph** n1 n2	*set graphics for phase portrait*
?	**skip** 10	*do not display 10 first time steps*
?	**border**	*hide axis*
?	**parameter** r 1 60 0.5	*vary r from 1 to 60 with increment 0.5*
?	**run** 1000	*display sort of movie*

savegraph 💾

abbreviation: **!**
syntax: **save** < $xxx.bmp$ >
parameters: $xxx.bmp$ name of bitmap file
function: Store graphic window in bitmap file `xxx.bmp`.
The index of the graphic window to be stored
can be specified using the **window** command.
The `bmp` file can be later modified, converted to `jpg` or printed.

- The **savegraph** command is useful in command files.
- Use alternatively the *File* | *Save* option 💾 in each graphic window.

Note In case of superimposed graphics (**addgraph** command), graphics are saved using a fixed window size.

Scatter graph

abbreviation: **j**
syntax: **scatter** $x\ y1 \ldots yN$
parameters: $x\ y1 \ldots yN$ names of variables
function: Display scatter plot of variables $y1, \ldots, yN$ as a function of variable x in graphic window, together with regression line.

- Use alternatively options *Scatter* and *Regress* in graphic windows *Settings*.
- The **scatter** command operates on window #1 unless window #i has been selected using the **window** command.

Example file `met_esd.ulm`

?	**change** ii 1	*set immigration indicator*
?	**newvar** er extratef(n)	*create new variable* er *equal to extinction rate of* n
?	**newvar** cvz cvzf(n)	*create new variable* cvz *equal to* *coefficient of variation of* n *with zeros excluded*
?	**scatter** cvz er	*parameterize scatter plot: relation variation/extinction*
?	**window** 2	*create and select graphic window #2*
?	**graph** t n	*parameterize graphic window #2*
?	**run** 10000 1000	*run model* *display scatter plot of variation/extinction along time in graphic window #1* *display population trajectory in graphic window #2*
?	**montecarlo** 1000 10000	*run Monte Carlo simulation (1000 trajectories)* *display scatter plot of variation/extinction at time 1000 in graphic window #1* *display average population trajectory in graphic window #2* *In status bar of graphic window #1 appears:* *a = 0.1315 b = -0.07304* *In main window appears:* *Regression: slope = 0.1315 intercept = -0.07304*

Sensitivity

abbreviation: **s**
other name: **sens**
remarks: Computations are not always feasible.
Does not work for relation-type models.

Usage 1:

syntax: **sens** < mat >
parameters: mat name of a matrix (default first matrix)
function: Give sensitivities and elasticities of the dominant eigenvalue λ
of matrix mat to changes in matrix entries.
When the matrix is not constant, sensitivities of the actual matrix
(at current time t) are given.
If the matrix is random or density dependent,
stochastic sensitivities are computed over 100 time steps.

- Select option *Matrix* | *Sensitivities* to access to the *Sensitivities* window.
- Select option *Matrix* | *Stochastic sensitivities* to access to the *Stochastic sensitivities* window, where the number of time steps can be parameterized.

Example 1 file `pass_0.ulm`

?	**sensitivity**	*sensitivities and elasticities of* λ *to changes in matrix entries*
?	**change** s0 beta1f(0.2,0.15)	*make matrix stochastic*
?	**sensitivity**	*give stochastic sensitivities*

Usage 2:

syntax: **sens** < mat > var
parameters: mat name of a matrix (default first matrix)
var name of a variable
function: Give sensitivities and elasticities of the dominant eigenvalue
of matrix mat to changes in variable var.
This is done via formal derivation.

Example 2 file `pass_0.ulm`

?	**sensitivity** s	*sensitivities and elasticities of λ to changes in s*
?	**sensitivity** v	*sensitivities and elasticities of λ to changes in v*
?	**sensitivity** f1	*sensitivities and elasticities of λ to changes in $f1$*
?	**sensitivity** s0	*s0 is the most sensitive parameter*
?	**change** s0 beta1f(0.2,0.15)	*make matrix stochastic*
?	**sensitivity** s0	*give sensitivity of actual λ to changes in s0* *+ stochastic sensitivity to changes in s0*

Usage 3:

syntax:	**sens** $< mat >$ $varx$ $vary$
parameters:	mat name of a matrix (default first matrix) $varx$ $vary$ names of 2 variables
function:	Display fitness landscape associated with variables $varx$ and $vary$. The matrix must be constant.

- Select option *Matrix | Landscape* to get the *Landscape* window, where the graphic bounds can be parameterized.

Example 3 file `pass_0.ulm`

- Select the *Matrix | Landscape* option. Provide s as *Variable X*, v as *Variable Y*. Click *Exec*. The λ-isoclines are drawn using the bounds $[0.5s, 1.5s]$ in X, and the bounds $[0.5v, 1.5v]$ in Y (the default). The isocline of the actual λ is drawn in red.
- Provide $s0$ as *Variable X*, $f1$ as *Variable Y*. Set the bounds in X to $[0, 1]$. Set the bounds in Y to $[0, 15]$. Click *Exec*.

Skip graph

abbreviation:	>
syntax:	**skip** T
parameters:	T integer > 0, number of time steps
function:	When the **run** command is executed, the first T time steps ($t = 0, \ldots, T-1$) are not displayed in the graphic window (but computed). Useful for bifurcation diagrams (see command **run**).

- Click the *Settings* button in the graphic window. The *Skip* field is in the *General* panel.
- Graphic window #i can be selected using the **window** command.

Spectrum graph

abbreviation:	**k**
other name:	**spec**
syntax:	**spec** *var*
parameters:	*var* name of variable
function:	Display the power spectrum of variable *var* in graphic window #1: decimal logarithms of square modulus of normalized frequencies $[0, 0.5]$. The system is run for 1024 time steps.

- Select the *Tools* | *Spectrum* option to get the *Spectrum* window, in which the number of time steps (a power of 2) can be parameterized.

Example file `regis.ulm`

?	**change** r 60	*display power spectrum with incommensurable discrete frequencies (quasi-periodicity)*
?	**spec** n	
?	**change** r 110	
?	**spec** n	*display continuous power spectrum (chaos)*

Text ▦ on/off

abbreviation:	**t**
syntax:	**text** $var1 \ldots varN$
parameters:	$var1 \ldots varN$ name of variables
function:	If *on*, display values of variables $var1, \ldots, varN$ in the main window, as the **run** or **montecarlo** command is executed.
default:	*on*

- Button ▦ allows to open up to 6 text windows, numbered #1 to #6, which can be parameterized using the *Settings* option ✕.
- Button · clears the *Text* window.
- For the Monte Carlo simulation, no more than 10000 rows can be displayed. The *Sampling interval* should be adjusted in accordance with the number of time steps.
- The text windows can be saved using the *File | Save option* 💾.
- Command **text** is totally independent of the text windows, though the purpose is similar.

Example 1 file `regis.ulm`

- Click ▦ to open text window #1, then ▶ to run the model 100 time steps. Values of time t and variables $n1$, $n2$ are displayed in the text window (every 10 time steps).
- Click the Settings option ✕, change $n1$ to n, remove $n2$. Change *Sampling interval* to 20. Click OK.
- Click ▶ to run the model 100 more time steps. Values of time t and variable n are displayed every 20 time steps.

Example 1 file `pass_2s.ulm`

- Select option *Montecarlo | Settings*. Change *Number of trajectories* to 1000 (*Number of time steps* is 50 by default). Click OK.
- Click ▦ to open text window #1. Click the *Settings* option ✕, change $nm1$ to n, remove $nm2$, $nf1$, $nf2$. Click OK.
- Click 🎲 to run the Monte Carlo simulation. Values of time t , mean population size (n) with standard error (SE), mean population size over non extinct trajectories (n^*) with standard error (SE) are displayed in the text window.

View 👁

abbreviation:	**v**
syntax:	**view** $o1\ldots oN$
parameters:	$o1\ldots oN$ name of ULM objects (matrix, vector, relation, variable, function)
function:	Display initial and actual values of objects $o1,\ldots,oN$.
syntax:	**view**
function:	Display all ULM objects.

- Button 👁 provides the *Variables* window with initial values, actual values and expressions of all variables. Expressions can be modified by selecting the corresponding field, modifying the expression, and typing `<return>`.
- The variables can be listed in *Evaluation order* or *Alphabetical order* (see bottom of the window).
- Option *Variables* | *All* lists all ULM objects as a hierarchical tree.
- Option *Variables* | *Calculator* is a desk calculator allowing the computation of mathematical expressions possibly involving ULM variables.

Example file `regis.ulm`

- Click 💡. Click button 👁. Change the expression of r to 30. Type `<return>`. Select option *Variables* | *All* to get the *Object*s window. The initial value of the matrix is shown in the right panel.
- Click ▶ to run the model 50 time steps. The trajectories stabilize. Select the *Objects* window. The panel is updated with the actual value of the matrix.
- Click ▶ to run the model 50 more time steps. Select the *Objects* window. It is checked that the matrix is almost constant.
- Click the *Property* button P to check that λ is 1.

Window on/off graph

abbreviation:	**w**
syntax:	**window** *i*
parameter:	*i* integer in $\{1,\ldots,6\}$, refer to graphic window #i
function:	Select or create graphic window #i, to which will apply all subsequent graphic commands: **addgraph** **border** **distribution** **erase** **line** **graph** **scatter** **savegraph** **xscale** **yscale**
default:	$i = 1$

- Useful to store several graphic windows using command files.

Example file `regis.ulm`

```
? graph t n                  parameterize graphics for window #1 (default)
? window 2                   create and select graphic window #2
  Graphic window #2 selected
? graph n1 n2                parameterize graphics for window #2
? run 500                    run model
                             display population trajectory in graphic window #1
                             display attractor in graphic window #2
```

Xscale on/off graph

abbreviation:	**x**
syntax:	**xscale** < $xmin$ > < $xmax$ >
parameter:	$xmin$ $xmax$ real numbers, bounds of graphics on the X axis
function:	Fix bounds $xmin$ and $xmax$ for abscissas (default: actual values).
default:	*off* (automatic scaling on the X axis)
see also:	**yscale**, **addgraph**

- Use alternatively option *Fix Xscale* in graphic windows *Settings*.
- Graphic window #i can be selected using the **window** command.

Yscale on/off graph

abbreviation:	**y**
syntax:	**yscale** <*ymin*> <*ymax*>
parameter:	$ymin$ $ymax$ real numbers, bounds of graphics on the Y axis
function:	Fix bounds $ymin$ and $ymax$ for abscissas (default: actual values).
default:	*off* (automatic scaling on the Y axis)
see also:	**xscale**, **addgraph**

- Use alternatively option *Fix Xscale* in graphic windows *Settings*.
- Graphic window #i can be selected using the **window** command.

6 MATHEMATICAL FUNCTIONS

6.1 Binary operators

The following binary operators are available:

- Usual arithmetical operators: $+ \ast / -$^(power)
- \ real modulo: $a \setminus b = a - b \ast \mathrm{trunc}(a/b)$. Examples: 7.4 \ 2 = 1.4, 7 \ 2 = 1
- @ convolution operator: F @ n = sum of n samples of distribution F. Examples: ber(p) @ n = binomf(n,p), poisson(f) @ n = poissonf(n,f).
- <: $a < b$ is 1 if a is strictly less than b, 0 otherwise
- >: $a > b$ is 1 if a is strictly greater than b, 0 otherwise

6.2 Unary operators

-	minus
sqrt	square root
abs	absolute value
trunc	integer part
	trunc(3.5) = 3, trunc(3.8) = 3, trunc(-3.5) = -4, trunc(-3.8) = -4
round	nearest integer
	round(3.2) = 3, round(3.6) = 4, round(-3.2) = -3, round(-3.6) = -4
ln	neperian logarithm
ln0	neperian logarithm extended to 0 by ln0(0) = 0
log	decimal logarithm
exp	exponential
fact	factorial
cos	cosinus
sin	sinus
tan	tangent
acos	inverse cosinus
asin	inverse sinus
atan	inverse tangent

6.3 Other operators

min	min($a_1,\dots,a_n$): minimum of the a_i' s
max	max($a_1,\dots,a_n$): maximum of the a_i' s
stepf	stepf(x,a,b) = 1 if $a \leqslant$ value of x $\leqslant b$, 0 otherwise
if	conditional: if(A,B,C) $\equiv$ if $A \neq 0$ then B else C if(2 < 3,1,2) = 1, if(trunc(2.5)-2,1,2) = 2
bicof	bicof(n,p): binomial coefficient $\binom{n}{p}$
lambdaf	lambdaf(i,j): modulus of j-th eigenvalue of i-th model lambdaf(1,1) = dominant eigenvalue of first model
prefv	prevf(x,k): previous value of variable x, k time steps backward prevf(x, 1) = value of x at the previous time step
bdf	bdf(n,b,d,Δ): discrete version of continuous birth-death process, similar to the poissonf function (see integer distributions below) n = number of individuals, b = birth rate, d = death rate, Δ = step of integration (choose $\Delta \approx 1/(b+d)$) n' = bdf(n,b,d,Δ) is similar to n' = poissonf(n, exp($b-d$))

6.4 Analysis of time series

We denote H the extinction threshold (default $H = 1$, see **montecarlo** command).

gratef	gratef(x): growth rate of variable x at time T : $\exp\left[\frac{\ln(x(T))-\ln(x(0))}{T}\right]$
textf	textf(x): extinction time of variable x, first time T such that $x < H$
meanf	meanf(x): average value $\bar{x}$ of variable x along time
variancef	variancef(x): variance VAR(x) of variable x along time
skewnessf	skewnessf(x): skewness $\gamma_1(x)$ of variable x along time
cvf	cvf(x): coefficient of variation of variable x = CV(x) = $\frac{\sqrt{VAR(x)}}{\bar{x}}$
meanzf	meanzf(x): average value of variable x with zeros excluded, values of x such that $x < H$
variancezf	variancezf(x): variance of variable x with zeros excluded, values of x such that $x < H$ are excluded
cvzf	cvzf(x): coefficient of variation of variable x with zeros excluded, values of x such that $x < H$ are excluded
nzf	nzf(x): number of zero values of variable x, number of dates τ such that $x(\tau) < H$
nef	nef(x): number of extinctions of variable x, number of dates $\tau \leqslant t$ such that $x(\tau-1) \geqslant H$ and $x(\tau) < H$
nif	nif(x): number of immigrations of variable x, number of dates $\tau \leqslant t$ such that $x(\tau-1) < H$ and $x(\tau) \geqslant H$
extratef	extratef(x): extinction rate of variable x, estimated at time t as ER(x) = $\frac{e}{t+1-z}$ if $x(t) < H$, and ER(x) = $\frac{e}{t-z}$ otherwise, with z = nzf(x) and e = nef(x), the number of dates $\tau \leqslant t$ such that $x(\tau-1) \geqslant H$ and $x(\tau) < H$
immratef	immratef(x): immigration rate of variable x, estimated at time t as IR(x) = $\frac{i}{z-i}$ if $x(t) < H$, and IR(x) = $\frac{i}{z}$ otherwise, with z = nzf(x) and i = nif(x), the number of dates $\tau \leqslant t$ such that $x(\tau-1) < H$ and $x(\tau) \geqslant H$

6.5 Random functions: continuous distributions

rand	rand(a)	uniform distribution over $[0, a]$
	domain	$a > 0$
	range	$[0, a]$
	mean	$a/2$
	variance	$a^2/12$
	density	$\frac{1}{a}\times$ characteristic function of $[0, a]$
gaussf	gauss(m, s)	Gaussian distribution mean m, standard deviation s
	domain	$s > 0$
	range	$\mathbb{R}$
	density	$\frac{1}{s\sqrt{2\pi}} \exp\left[-\frac{1}{2}\frac{(x-m)^2}{s}\right]$
gauss	gauss(s)	Gaussian distribution mean 0, standard deviation s gauss(s) = gaussf($0, s$)
gamm	gamm(a)	gamma distribution with parameter a
	domain	$a > 0$
	range	$\mathbb{R}_+^*$
	mean	a
	variance	a
	density	$\frac{1}{\Gamma(a)} x^{a-1} e^{-x}$
betaf	betaf(a, b)	beta distribution with parameters a, b
	domain	$a > 0$, $b > 0$
	range	$[0, 1]$
	mean	$\frac{a}{a+b}$
	variance	$\frac{ab}{(a+b+1)(a+b)^2}$
	density	$\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x e^{a-1} e^{-x}$

beta1f	beta1f(m, s)	variant of beta distribution mean m, standard deviation s
	domain	$m > 0$, $0 < s^2 < m(1 - m)$
	range	$[0, 1]$
	mean	m
	variance	s^2
	remark	the distribution is bell-shaped for small s and U-shaped for large s
expo	expo(a)	exponential distribution with parameter a
	domain	$a > 0$
	range	$\mathbb{R}_+^*$
	mean	$\frac{1}{a}$
	variance	$\frac{1}{a^2}$
	density	$a \exp(-ax)$
lognormf	lognormf(m, s)	lognormal distribution mean m, standard deviation s
	domain	$m > 0$, $s > 0$
	range	$\mathbb{R}_+^*$

6.6 Random functions: integer distributions

ber	ber(p)	Bernoulli samples
		$\mathbb{P}(X = 0) = 1 - p$, $\mathbb{P}(X = 1) = p$
	domain	$0 \leqslant p \leqslant 1$
	range	$\{0, 1\}$
	mean	p
	variance	$p(1 - p)$
	generating function	$f(s) = (1 - p) + ps$
binomf	binomf(n, p)	Binomial distribution
		$\mathbb{P}(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$
	domain	$n \geqslant 0$, $0 \leqslant p \leqslant 1$
	range	$\{0, 1, \ldots, n\}$
	mean	np
	variance	$np(1 - p)$
	generating function	$f(s) = ((1 - p) + ps)^n$
nbinomf	nbinomf(r, p)	negative binomial distribution
		$\mathbb{P}(X = k) = \binom{k+r-1}{r-1} p^r (1 - p)^k$
	domain	r real, $0 \leqslant p \leqslant 1$
	range	$\mathbb{N}$
	mean	$r(1 - p)/p$
	variance	$r(1 - p)/p^2$
nbinom1f	nbinom1f(m, s)	negative binomial distribution
		mean m, standard deviation s
	domain	$0 < m < s^2$
	range	$\mathbb{N}$
	mean	m
	variance	s^2

poisson	*poisson(m)*	Poisson distribution with mean m
		$\mathbb{P}(X=k)=e^{-m}\frac{m^k}{k!}$
	domain	$m \geqslant 0$
	range	$\mathbb{N}$
	mean	m
	variance	m
	generating function	$f(s)=\exp(m(s-1))$
poissonf	poissonf(n,m) =	sum of n samples of poisson(m)
geom	geom(p)	geometric distribution with parameter p
		$\mathbb{P}(X=k)=p(1-p)^k$
	domain	$0 \leqslant p \leqslant 1$
	range	$\mathbb{N}$
	mean	$\frac{1-p}{p}$
	variance	$\frac{1-p}{p^2}$
	generating function	$f(s)=\frac{p}{1-(1-p)s}$
tabf	tabf$(p_0,\ldots,p_n)$	tabulated distribution
		$\mathbb{P}(X=k)=$ if $k \leqslant n$ then p_k else 0
	domain	$0 \leqslant p_k \leqslant 1$, $p_0+\cdots+p_n=1$
	range	$\{0,1,\ldots,n\}$
	mean	$m=f'(1)=p_1+2p_2+\cdots+np_n$
	variance	$f''(1)+m-m^2$
	generating function	$f(s)=p_0+p_1s+\cdots+p_ns^n$

7 TECHNICAL NOTICE

7.1 Specifications

Computer	PC, MAC
System	Windows®, Linux, macOS®
Minimal memory required	1 Go
Programming language	Object Pascal – Borland Delphi 6 Compiled under Free Pascal/Lazarus
Source code size	~ 17000 lines
Distribution package size	~ 5 M

7.2 Program bounds

General	
maximum number of models in the same model file	5
maximum size of models (size of matrix or number of relations)	100
maximum number of relations (total)	500
maximum number of variables	5000
Graphics	
maximum number of graphics windows	6
maximum number of trajectories per window	4
best graphic resolution in number of time steps	10000
Text	
maximum number of text windows	6
maximum number of variables per window	16 (4 for Monte Carlo)
maximum number of lines per window	10000 for Monte Carlo
File	
maximum number of output text files	5
maximum number of variables per file	10

8 ULM DISTRIBUTIONS

8.1 Downloads

ULM web page `https://www.biologie.ens.fr/~legendre/ulm/ulm.html`

Computer/System	Download	Install
PC Windows® 64-bit	Compressed archive `ulm.zip`	Program file `ulm.exe`

PC Linux
64-bit
	Compressed archive
	`ulm.tar.gz`
	Expand using command
	`tar -xzf ulm.tar.gz`
		Program file
		`ulm`

macOS® 64-bit	Compressed package `ulm.dmg`	Program file `ulm`

All distributions also contain a console (no graphics) version: `ulmc`.

8.2 Source files

The ULM source files and compiling facilities for Windows®, Linux and macOS® are provided under the GitLab environment:

GitLab site `https://gitlab.com/ecoevomath/ulm`

8.3 Acknowledgements

The developers of Free Pascal/Lazarus are gratefully acknowledged for their wonderful work. Special thanks to Karl-Michael Schindler for help on the macOS® distribution.

REFERENCES

[Barot et al., 2002] Barot, S., Gignoux, J., and Legendre, S. (2002). Stage-classified matrix models and age estimates. *Oikos*, 96:56–61.

[Beissinger and McCullough, 2002] Beissinger, S. R. and McCullough, D. R. (2002). *Population Viability Analysis*. University of Chicago Press.

[Beissinger and Westphal, 1998] Beissinger, S. R. and Westphal, M. I. (1998). On the use of demographic models of population viability in endangered species management. *The Journal of Wildlife Management*, 32:821–841.

[Bessa-Gomes et al., 2003] Bessa-Gomes, C., Danek-Gontard, M., Cassey, P., Møller, A. P., Legendre, S., and Clobert, J. (2003). Mating behaviour influences extinction risk: insights from demographic modelling and comparative analysis of avian extinction risk. In *Annales Zoologici Fennici*, pages 231–245. JSTOR.

[Bienvenu and Legendre, 2015] Bienvenu, F. and Legendre, S. (2015). A new approach to the generation time in matrix population models. *The American Naturalist*, 185:834–843.

[Caswell, 1989] Caswell, H. (1989). *Matrix Population Models*. Sinauer Associates, Sunderland, MA.

[Caswell, 2001] Caswell, H. (2001). *Matrix Population Models – Construction, Analysis, and Interpretation*. Sinauer Associates, Sunderland, MA, 2nd edition.

[Caswell and Weeks, 1986] Caswell, H. and Weeks, D. E. (1986). Two-sex models: chaos, extinction, and other dynamic consequences of sex. *The American Naturalist*, 128:707–735.

[Cazelles and Ferrière, 1992] Cazelles, B. and Ferrière, R. (1992). How predictable is chaos? *Nature*, 355:26.

[Costantino et al., 1995] Costantino, R. F., Cushing, J. M., Dennis, B., Desharnais, R. A., et al. (1995). Experimentally induced transitions in the dynamic behaviour of insect populations. *Nature*, 375:227–230.

[Courchamp et al., 1999] Courchamp, F., Clutton-Brock, T., and Grenfell, B. (1999). Inverse density dependence and the allee effect. *Trends in Ecology & Evolution*, 14:405–410.

[E. and S Ellner, 1992] E., C. M. and S Ellner, S. (1992). Simple methods for calculating age-specific life history parameters from stage-structured models. *Ecological Monographs*, 62:345–364.

[Ferrière and Gatto, 1995] Ferrière, R. and Gatto, M. (1995). Lyapunov exponents and the mathematics of invasion in oscillatory or chaotic populations. *Theoretical Population Biology*, 48:126–171.

[Ferrière et al., 1996] Ferrière, R., Sarrazin, F., Legendre, S., and Baron, J. (1996). Matrix population models applied to viability analysis and conservation: theory and practice using the ULM software. *Acta Œcologica*, 17:629–656.

[Gosselin and Lebreton, 2000] Gosselin, F. and Lebreton, J.-D. (2000). Potential of branching processes as a modeling tool for conservation biology. In *Quantitative Methods for Conservation Biology*, pages 199–225. Springer.

[Griffith et al., 1989] Griffith, B., Scott, M. J., Carpenter, J. W., and Reed, C. (1989). Translocation as a species conservation tool: status and strategy. *Science*, 245:477–480.

[Grimm, 1999] Grimm, V. (1999). Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future? *Ecological modelling*, 115:129–148.

[Houllier and Lebreton, 1986] Houllier, F. and Lebreton, J.-D. (1986). A renewal-equation approach to the dynamics of stage-grouped populations. *Mathematical Biosciences*, 79:185–197.

[Inchausti and Halley, 2002] Inchausti, P. and Halley, J. (2002). The long-term temporal variability and spectral colour of animal populations. *Evolutionary Ecology Research*, 4:1033–1048.

[Kokko and Ebenhard, 1996] Kokko, H. and Ebenhard, T. (1996). Measuring the strength of demographic stochasticity. *Journal of Theoretical Biology*, 183:169–178.

[Lebreton, 1996] Lebreton, J.-D. (1996). Demographic models for subdivided populations: the renewal equation approach. *Theoretical Population Biology*, 49:291–313.

[Legendre, 2004] Legendre, S. (2004). Age structure, mating system and population viability. In *Evolutionary conservation biology*, pages 41–58. Cambridge University Press, Cambridge.

[Legendre and Clobert, 1995] Legendre, S. and Clobert, J. (1995). ULM, a software for conservation and evolutionary biologists. *Journal of Applied Statistics*, 22:817–834.

[Legendre et al., 1999] Legendre, S., Clobert, J., Møller, A. P., and Sorci, G. (1999). Demographic stochasticity and social mating system in the process of extinction of small populations: the case of passerines introduced to New Zealand. *The American Naturalist*, 153:449–463.

[Lundberg et al., 2000] Lundberg, P., Ranta, E., Ripa, J., and Kaitala, V. (2000). Population variability in space and time. *Trends in Ecology & Evolution*, 15:460–464.

[Mills et al., 1996] Mills, L. S., Hayes, S. G., Baldwin, C., Wisdom, M. J., Citta, J., Mattson, D. J., and Murphy, K. (1996). Factors leading to different viability predictions for a grizzly bear data set. *Conservation Biology*, 10:863–873.

[Møller and Legendre, 2001] Møller, A. P. and Legendre, S. (2001). Allee effect, sexual selection and demographic stochasticity. *Oikos*, 92:27–34.

[Possingham et al., 2002] Possingham, H. P., Lindenmayer, D. B., and Tuck, G. N. (2002). *Population viability analysis*, chapter Decision Theory for Population Viability Analysis, pages 470–489. University of Chicago Press, Chicago, Illinois.

[Sarrazin and Legendre, 2000] Sarrazin, F. and Legendre, S. (2000). Demographic approach to releasing adults versus young in reintroductions. *Conservation Biology*, 14:488–500.

[Schoener et al., 2003] Schoener, T. W., Clobert, J., Legendre, S., and Spiller, D. A. (2003). Life-history models of extinction: a test with island spiders. *The American Naturalist*, 162:558–573.

[Stephens and Sutherland, 1999] Stephens, P. A. and Sutherland, W. J. (1999). Consequences of the allee effect for behaviour, ecology and conservation. *Trends in Ecology & Evolution*, 14:401–405.

[Thomas-Orillard and Legendre, 1996] Thomas-Orillard, M. and Legendre, S. (1996). Virus C de la drosophile et dynamique d'une population hôte. *Comptes Rendus de l'Académie des Sciences de Paris*, 319:615–621.

[Tuljapurkar, 1990] Tuljapurkar, S. (1990). *Population Dynamics in Variable Environments*, volume 85. Springer Science.