PISCATOR 2.0

Model description

Egbert van Nes

March, 1999

Contents

1. Introduction

Piscator aims at predicting the fish stock given a specific fishery scenario, and the spin-off on water quality and nature conservancy value by indirect effects on zooplankton, mussels and benthic organisms. It is a multi-species individual-based model. Such model of the dynamics of interacting populations is notoriously complex and using it can be problematic.

1.1 How to deal with complexity?

Piscator aims at predicting the fish stock given a specific fishery scenario, and the spin-off on water quality and nature conservancy value by indirect effects on zooplankton, mussels and benthic organisms. However, modelling the dynamics of interacting populations is notoriously problematic.

One of our major concerns is to control the complexity, because it is essential that we obtain the right results for the right reasons. To minimise complexity we:

1. use a flexible design
2. shut down feedback mechanisms
3. simplify poorly known processes

1.2 Flexible design

Object oriented programming (OOP) facilitates the design of a flexible program. Fish species, nets, birds, zooplankton, zoobenthos can be added or removed from the model without changing the program code. Even new species (with different parameter settings) can be defined. By adding or removing 'inhabitants' from the model the complexity can be changed stepwise. Furthermore, fish is represented by 'super-individuals', each individual has an additional feature, namely the number of individuals that it represents. Obviously this approach can cause loss in variation. By increasing the number of super-individuals per year class, the variation can be increased to focus on complex processes like the competitive bottle-neck.

1.2.1 The super-individual

A general problem with individual based models is the large number of individuals that is required. This is necessary to avoid loss of variation, irregular dynamics, and large sensitivity to the value of random generator seeds. These large numbers of individuals result in impractical large computation times. We follow the practical *super-individual* approach. That means that each model individual has one extra feature: the amount of population individuals that it actually represents. Mortality for example is realised in individual based models as the probability of dying each time step. This is done by drawing a random number and deciding

from the value of this number to die or not to die (=drawing from a binomial distribution with k=1). In super-individuals this can be done repetitively, while subtracting the number of dead individuals from the internal amount. This is equivalent to drawing from a binomial distribution, which is computationally more convenient.
In the present model we use discrete-event simulation. Discrete-event simulation is an efficient way of handling irregular actions, but if there are many regular events, it is inefficient. Therefore, we use (optionally) a combination of traditional Euler integration and discrete event simulation.

1.2.2 Pandora's box

In the interaction between fish species there are two important links:
1. Mortality of prey fish depends on consumption by predators (including fishery).
2. Growth and survival of predatory fishes depends on the availability of prey fish.

Together these links form a feedback mechanism, which is notoriously complex in its behaviour. The feedback mechanism can be shut off, by replacing the interactions by a fixed value, as is done in one species models.

We designed a stepwise approach to open carefully the Pandora's box of complex model behaviour, by implementing two switches for each species that can set both links on or off. The part of mortality caused by fisheries and by fish or bird predation can be replaced by a fixed (size dependent) mortality. The daily deviation of this fixed mortality rate from the actual computed mortality by predation can be studied in the 'mortality table'.

Growth is affected by temperature, body size and food uptake. The food dependent part of the consumption is modelled using a saturating type of functional response accounting for selectivity for different available prey sizes and types. As an alternative a fixed value is assigned to the functional response (e.g. 0.7). Again the daily deviation is represented in the mortality table.

The switches can be set for each species independently. During the calibration of the model the four possible combinations are:

Step 1: fixed mortality; fixed growth
Step 2: fixed mortality; food dependent growth
Step 3: mortality influenced by predation; fixed growth
Step 4: mortality influenced by predation; food dependent growth

Step 1. Fixed mortality; fixed growth
The growth depends only on temperature, and not on available food. The fish mortality is not affected by predation, but has a fixed value.

The fixed values for size dependent mortality and functional response are tuned for each species to fit the frequency distribution of year classes and the individual weights in each year class to the observations. At this point the model is a straightforward Leslie-matrix instrument. By skipping fisheries from the fixed mortality simple calculations can be made to estimate the direct effects of fisheries on the structure of the populations. Indirect effects due to shifts in predation and competition are still excluded.

Step 2. Fixed mortality; Food dependent growth
The fixed value for the functional response of piscivores is substituted by a saturating function of available food corrected for gape limitation and specific prey preferences. It is crucial that the prey selectivity is set realistically using data of gut contents in relation to available prey. The half saturation prey density in the type-II functional response is chosen to fit observed individual growth at corresponding prey densities and temperatures.
The model can now be used to compute the effects of different prey densities on the growth of piscivores. Since mortality is still fixed, no feedback loops exist.

Step 3. Dependent mortality; fixed growth
After shutting off the growth dependence on food, we use the functional response developed in step 2 to consume and remove preys. Since this accounts for part of the former fixed mortality, the background mortality is lowered to keep overall mortality in accordance with observed dynamics in the field.
At this point cascading effects of fisheries can be computed (fishery -> pikeperch -> smelt). Since growth is fixed, feed-back loops are absent. Effects of food availability on piscivores, and consequently competition for food can not be studied.

Step 4. Dependent mortality; dependent growth
The model can now be used to study the complete dynamic response of the interacting fish community to different fishery scenarios.

1.2.3 Dealing with poorly known processes

Eggs and larvae.
We simply plug in a fixed number of y-o-y fish at the first of June. This is the best we can do, since variations in recruitment are not related to spawning stock in IJsselmeer.

Interactions of fish and plankton community
As a start we will assume a constant planktonic food availability. Later this will be substituted by simple logistically growing food, than can be depleted, causing food limitation of planktivores.

Migration and distribution patterns
Since the processes responsible for the distribution of predators and prey are not yet sufficiently documented we will ignore these processes.

1.3 Implementation of the model

We programmed in the object-oriented programming style (OOP). As language we used Delphi version 4 (Object Pascal). As host system we use a PC with Microsoft Windows 95.

1.3.1 Object-oriented programming

Object-oriented programming (OOP) is a modern extension of structured programming. In structured programming parts of the program are defined as modules (called subroutines, procedures or functions), that get data from the main program. All data (except temporary data) are stored in the main program. New in object-oriented programming is that program modules (called objects) are defined as a combination of data and procedures. Each object is an independent part of the program. The objects can communicate by calling each others' procedures (called methods).
The definition of the objects is organised in a hierarchy. This hierarchy is based on two special features of the objects: inheritance and polymorphism. Each object inherits all properties (data and procedures) of its parent, and can add new properties to this. It can also change some properties of its parent, by overruling part of the inherited procedures. This feature (polymorphism) makes OOP very flexible, since it makes it possible to reuse program code for other purposes than it was originally designed for.
In text books about OOP the analogy with biology is often used (a beetle inherits properties from an insect, which is a special case of an animal, etc.). Obviously, OOP is very suitable for individual based models and it seems almost as if OOP is special designed for individual based models.

1.3.2 The structure of the program

The program is divided into two parts, (1) the graphical user interface (GUI) and (2) the model. This separation of program code is also called the Model-View-Controller concept. When the model changes the views are notified of the type of change. The Controller 'talks' to the user.

(1) The graphical user interface
An essential part of the development of the model is the design of a user-friendly interface. It should be possible for the user to change all parameters, and to view model results in different types of graphs. An on-line help system should provide the user with case-sensitive help. Under Microsoft Windows (especially when combined with Borland objects) it is fairly easy to create simple user interaction by standardised menus, dialog windows, list boxes. The model outputs can be shown in multiple graph windows.

(2) The model
Three collections of objects are defined:

(a) a collection of addresses and names of parameters that can be changed by the user (the user interface uses this information to change parameters);
(b) a collection of inhabitants (objects that are interacting: nets, fishes, birds, etc.);
(c) a collection of samplers (objects that collect model results from the inhabitants and notify the graphical user interface each time there are new results).

Each time step all inhabitants and samplers are called for action. All inhabitants and samplers are represented by objects with the same ancestor and can be treated in the same way. Due to polymorphism of the objects, the resulting actions can be very different.

1.4 Analysers

Several flexible types of special analyses are available for more elaborate studies of model behaviour:

Parameter analysis - change parameters in a structured way
Monte-Carlo sensitivity analyser - sensitivity of the model to parameter change
Bayesian uncertainty analyser - uncertainty of the model outcome
Parameter calibration CRS - automatic calibration
Equilibrium analyser - study the model in equilibrium
Stochasticy analyser - rerun the model to detect stochastic mechanisms

1.4.1 Parameter analyser

Change one or few parameters gradually and plot the effects on the results. A special version of this analysis can be used to detect alternative stable states. A conditioning parameter that causes the shift to an alternative stable state is first gradually increased and after the shift decreased again.

The parameter *NStepsAnal* defines the number of steps to be taken after an initial stabilising period of *NInitStabilise* years.
There can be 3 different stages in each step:

1. Stabilise a period *NStabilise* years
2. Disturb the equilibrium (optional for studying resilience) *Ndisturbance* years
3. Write the results *NYears* years

The parameter *ParamArray* holds all information that is needed for each parameter to be changed (among others: min, max, nsteps, see: Parameter for analysis dialog) The parameter *ParamCount* determines the number of parameters to be changed. The parameter *ResetEachStep* determines the whether the model should be reset after each step.

RunType can be used to make the analyser inactive.

The parameter *Dummy* has no function in the model. It can be used as a dummy variable to be grouped with a parameter group to produce graphs with another scale than the parameters that were varied.

1.4.2 Monte-Carlo sensitivity analyser

This analyser does a sensitivity analysis like Klepper (1989). Change several parameters at random and independently. By analysing the impact of the changes on the results, the most critical parameters can be detected. Moreover the parameters can be clustered into groups that have the same or opposite effects on the model results.

The parameter *NStepsAnal* defines the number of steps to be taken after an initial stabilising period of *NInitStabilise* years.
There can be two different stages in each step:
1. Stabilise a period *NStabilise* years
2. Write the results *NYears* years

The parameter *ParamArray* holds all information that is needed for each parameter to be changed (among others: mean, standard deviation and distribution, see: Parameter of Monte-Carlo analysis dialog) The parameter *ParamCount* determines the number of parameters to be changed.

The parameters *ClusMethod* and *DistMeasure* determine how the cluster analysis is performed. These parameters can also be changed while creating a

1.4.3 Bayesian uncertainty analysis

Bayesian uncertainty analysis. The same as a Monte-Carlo sensitivity analysis, but with a different target. Use this analysis to estimate the uncertainty in the model output, by drawing all parameters from statistical distributions. The model results are presented as percentiles (5%, 25%, 50% 75% and 95%).

The parameter *NStepsAnal* defines the number of steps to be taken after an initial stabilising period of *NInitStabilise* years.
There can be 2 different stages in each step:
1. Stabilise a period *NStabilise* years
2. Write the results *NYears* years

The parameter *ParamArray* holds all information that is needed for each parameter to be changed (among others: mean, standard deviation and distribution, see: Parameter of Monte-Carlo analysis dialog) The parameter *ParamCount* determines the number of parameters to be changed.

1.4.4 Controlled Random Search (CRS) analyser

Controlled random search (Price, 1979; Klepper & Hendrix, 1994a; Klepper & Hendrix, 1994b). Calibration of one or more parameters. The user must provide one or more data files with data that are compatible with data generated by the time graphs. The parameters are varied within preset fixed boundaries. The average or minimal adjusted R2 of each data set is used as goodness of fit criterion. During the process the boundaries are narrowed to increase the efficiency. Note that the amount of computing time increases at least quadratic with the number of parameters considered (Klepper & Hendrix, 1994b). Therefore, the number of parameters that are calibrated simultaneously, should be kept low. De Hoop et al. (1992) describe the technical details of implementing the method in a clear way.

This analyser may use a number of initial stabilising years (usually not needed in the analysis) *NInitStabilise* and a number of stabilising years (*NStabilise*, not needed either).

NYears defines the number of years to be run (should correspond with the years with data). The parameters *ParamCount* and *ParamArray* define the parameters and their ranges in which they are changed. The parameter *NVase* defines the number of elements in the "vase". The higher this number the slower convergence is achieved, but the smaller the risk of sticking in a local optimum. The parameter *GofCrit* defines the convergence criterion. The model results are compared with data from files, see *SamplerFiles*. The parameter *ConvCrit* sets the convergence criterion ((best parameter setting - worst setting in vase)/best parameter setting), but this parameter can be overruled by the maximum number of steps *NSteps*.

1.4.5 Equilibrium analyser

Equilibrium analyser This simple analyser lets the model first stabilise for some years. Thereafter, the equilibrium state of the model is analysed.
After the initial stabilising period of *NInitStabilise* years, the program writes the results of *NYears* years

1.4.6 Stochasticy analyser

Stochasticy analyser - This simple analyser repeats a certain run many times to analyse the stochasticy of the model.
After an optional initial stabilising period of *NInitStabilise* years, a simulation run of *NYears* is repeated *NStepsAnal* times.

1.5 The structure of the model

The basic components of the model are:

Fishes
- maximum growth
- maximum consumption(temperature dependent)
- fish predation(functional responses, size/species selectivity, habitats)
- zooplankton/benthos feeding(functional responses, species selectivity)
- size dependent food switches
- fixed recruitment, weight loss during spawning
- mortality(background, starvation, predation or fixed mode mortality)

Fishery (fykes, gill-nets and seines)
- size/species selectivity
- spawning period (increased fyke catches)
- seasonal effort (monthly variations)

Birds
- size/species selectivity
- fixed numbers

Zooplankton
- fixed concentrations (or minimal model with phytoplankton)

Zoobenthos, zebra mussels
- fixed concentrations (or minimal model)

Environment
- temperature
- habitats

See also: complexity, analysers

2. Environment

2.1 Temperature

The temperature can be read from a file with daily water temperatures.
Alternatively, the temperature can be modelled with a cosine function:

$$T = \left[T_{\max} - \frac{T_{\max} - T_{\min}}{2} \left(1 + \cos\left(\frac{2\pi}{365.25} (day - T_{lag}) \right) \right) \right] T_{dev}$$

in which:

T	Temperature (°C)
T_{max}	Maximum temperature (°C) *TempMax*
T_{min}	Minimum temperature (°C) *TempMax*
T_{lag}	Temperature lag (°C) *TempMax*
T_{dev}	Temperature deviation factor (factor) *TempDev*
day	Number of days after the first of January (d)

The parameter *DataFile* determines if data are read from a file. The parameter should contain the file name or NAN if no files are read. The format of the data files is flexible.
The way that data are read from the file is determined by the following parameters:

I. *TempFileColumn* is the column that contains temperature. The first column always contains the dates. If the temperature data are in the second column then the parameter is 1.
II. *TempDrawOutside* determines what happens outside the scope of the data file. If this parameter is *False*, the cosine function is used outside the scope of the data file, else random years are drawn (with replication) from the data file. Only full years are drawn.
III. *TempPermuteMode* determines what happens inside the scope of the data file. This parameter can have three values:
    A. RealData - the real data are used and remain unchanged.
    B. PermuteYears - the years in the file are permuted (drawn without replication)
    C. DrawYears - the years are resampled with replication

If one file is used also for other parameters (in other columns), the file is only read once in the computer memory. Then, the same years are drawn for these parameters, so all resampling is then linked. If this behaviour is not wished, use copies of the data file.

2.2 Habitats

You can define a number of different habitats in the model. Within each habitat, predators see

only the preys that are in that habitat. The habitats are defined with the following parameters:
1. *NHabitat* defines the number of different habitats.
2. *HabitatArea* defines the relative area of each habitat. The sum of all habitat areas should be 1 (the last element of the array is automatically corrected if the sum is not 1)
3. *HabitatNames* is used to name the habitats (for example: littoral, pelagical etc.)

2.2.1 Habitats of fishes

If there are habitats selected, fishes can be distributed unevenly over the habitats. For different length classes, these distribution may be different. This is defined by two parameters:
1. *nHabitatChoices* defines the number of length classes with different habitat choices. If this parameter equals zero, the fish density is equally distributed.
2. *HabitatChoices* is an array with minimal length of each length class and the distribution of the fishes over the different habitats (see also: habitats). The sum of all habitat choices should be 1 (the last element of the array is automatically corrected if the sum is not 1). The fishes are all the time distributed over these habitats according to the habitat choices. It does not matter if the fishes grow faster in one of the habitats.

The distribution over the different habitats has in particular consequences for the food availability for the fishes. The fishes can only eat within each habitat. The food may be equally distributed over the habitats or not.

2.2.2 Habitat of zoobenthos, zooplankton, nets, birds, mussels and phytoplankton

All other species than fish can either be equally distributed over the habitats or not. The parameter *EqualDistr* determines whether the distribution is proportional to the area of the habitats or not. If this parameter is set to *False* then the user may enter a fixed distribution over the habitats (parameter *HabitatDistr*).

3. Fishes

3.1 Growth

The maximum growth and consumption is related to fish weight and temperature. The realised consumption is dependent on food availability too.

First the maximum growth is computed from fish weight and temperature. The parameters can be calibrated with field data.

Then the maximum consumption is calculated by estimating the respiration and the assimilation efficiency

If the food is sub-optimal only a part of the consumption can be realised. This is regulated by the functional response.

Finally the consumption is realised by lowering the food biomass, and the realised growth is calculated back.

3.1.1 Maximum growth

The growth of a fish at optimal food conditions is related to its weight and to temperature. For each species a growth curve is established according to the following equation:

$$(\frac{dW}{dt})_{max} = (G_{mean} + T_1 (T - T_0) + T_2 (T - T_0)^2) W^{b_c}$$

in which (names of parameters between brackets):

W	individual weight (g)
t	time (days)
G_{mean}	average growth of a one gram fish at T_0 °C (day^{-1}) *AvGrowth*
T_0	mean annual water temperature (in the Netherlands: 17.5 °C)
T	actual temperature (°C)
T_1	first order temperature effect ($day^{-1}°C^{-1}$) *T1*
T_2	optional second order temperature effect, at default T_2=0 (day^{-1} $°C^{-2}$) *T2*
b_c	exponent of ± 0.6 *b_cons*

The parameters G_{mean}, T_1 and b_c are fitted using field data using Controlled Random Search. The second order term (T_2) is usually negligible. It is mainly added as a check for non linearity during the optimising process.

3.1.2 Maximum consumption

To realise the maximum growth the fish has to consume food. The first step is to calculate the maximum consumption:

$$C_{max} = \frac{(\frac{dw}{dt})_{max} + R_{max}}{A_f}$$

in which

R_{tot}	respiration, a function of temperature and body weight (g day^{-1})
A_f	assimilation efficiency of the consumed food item(parameter *FoodSwitch*)
W	individual weight (g)
t	time (days)

see also: realised consumption

3.1.3 Respiration

For respiration we use a standard metabolism function. In the constant a_r the respiration due to

food consumption is included.

$$R_{\max} = a_r W^{b_r} Q_{10}^{\frac{T-T_r}{10}}$$

in which

T_r	reference temperature (20 °C) *TRefResp*
Q_{10}	temperature effect per 10 °C *Q10Resp*
T	temperature °C
a_r	constant *AResp*
b_r	exponent *b_resp*

3.1.4 Realised growth

If the food conditions are not optimal, only a part of the maximum consumption (p) can be realised. The realised growth is calculated back:

$$\frac{dW}{dt} = p\, C_{\max}\, A_f \text{ - } R_{tot}\, r_{growth}$$

p	part of the maximum consumption that is realised: constant in fixed consumption mode (*FuncResp*) or functional response
A_f	assimilation efficiency of the consumed food item (parameter *FoodSwitch*)
R_{tot}	respiration function, a function of temperature and body weight (g day^{-1})
r_{growth}	Respiration due to growth, function of functional response
C_{max}	maximum consumption, a function of temperature and body weight (g day^{-1})
W	individual weight (g)
t	time (days)

3.1.5 Respiration due to growth

When the consumption of fishes is low, the metabolism is reduced. We assume that the respiration decreases with the functional response to at most 1/3 of the maximum respiration:

$$r_{growth} = \frac{1}{3 - 2\frac{C}{C_{\max}}}$$

3.2 Weight-length function

If a fish grows, the new weight is converted to standard length by the allometric relation:

$$W = a\, L^b \quad or \quad L = \left(\frac{W}{a}\right)^{\frac{1}{b}}$$

W	individual weight (g)
L	standard length (cm)
a	constant (g cm^{-1}) *ALen*
b	exponent *BLen*

If the fish looses weight, the length is not changed. The deviation of the actual length from the standard length on basis of the actual weight is a measure of the condition of the fish (see also: starvation mortality).

3.3 Functional response

The functional response relates the consumption rate of an individual to the food density. The most commonly used functional response is the type 2 functional response, which is a saturating function.
The problem with the classical functional response is that the half saturation value is different for fishes with different length, because larger fishes swim faster, inspecting an larger volume. On the other hand, large fish need more food. Furthermore the available food biomass changes because large fishes can consume larger preys.

The volume of the lake that a fish can inspect is a function of swimming speed, reactive distance and temperature. Both swimming speed and reactive distance are assumed to be proportional with the fish length, so:

$$V = a_1 (a_2 L)^2 a_3 L \;\; A_T = a L^3 \;\; A_T$$

in which:

V	inspected volume (m^3)
L	predator length (cm)
A_T	unknown temperature function.

The available food biomass is the total available food biomass (which is a function of predator length) present in the volume that is inspected per day. If the lake volume is smaller than the inspected volume a smaller time step is required.

The consumption rate varies between 0 and maximum consumption C_{max} per day. To correct for different needs of small and large fishes, the available food biomass is divided by C_{max}. The quotient of V*TotFB and C_{max} can be simplified and approximated by assuming that the relation with temperature is the same for all components:

$$AvFood = \frac{TotFB\,V}{C_{\max}} = a\,\frac{TotFB\,L^3\,A_T}{B_T W^{b_r} + C_T W^{b_c}} \approx a\,TotFB\,\frac{W}{W^{2/3}} \approx a\,TotFB\,L \qquad 1$$

in which:

TotFB	Total available food biomass, can be function of predator length, prey length, prey species (kg ha^{-1})
AvFood	Available food biomass
B_T,C_T	temperature functions

Thus, we get the following functional response:

$$p = \frac{C_r}{C_{\max}} = \frac{AvFood}{AvFood + H_1} S = \frac{TotFB\ L}{TotFB\ L + H} S \qquad (2)$$

in which

C_r	realised consumption (kg ha^{-1} d^{-1})
S	suitability of the type of food (a function of predator length)
TotFB	Total available food biomass, can be function of predator length, prey length, prey species (kg ha^{-1} d^{-1})
H	half-saturation value (cm kg ha^{-1} d^{-1}) *SpecSelect*

3.3.1 Total available food biomass

For each food item (fish, zooplankton, zoobenthos, mussels) the total available food biomass must be determined accounting for the suitability of that specific prey characterised by a selectivity index:

$$TotFB = \sum_{j=0}^{n} \sum_{i=0}^{N_j} W_i\ Sel_{ij} \qquad (3)$$

in which:

n	number of species of food item (e.g. fishes) in the current habitat
N_j	number of individuals of species j
W_i	individual biomass (kg)
Sel_{ij}	selectivity of individual i of species j, function of own length and prey length and prey species (ranges between 0 and 1) *SpecSelect*.

3.3.2 Realising requested consumption

After the consumption is calculated, the requested food biomass must be subtracted from the prey populations. This is done in a second computational loop by giving each individual a probability (m_{ij}) of mortality:

$$m_{ij} = \frac{W_i \, Sel_{ij}}{TotFB}$$

4

TotFB	Total available food biomass, can be function of predator length, prey length, prey species (kg)
Sel_{ij}	selectivity of individual i of species j, function of own length and prey length and prey species (ranges between 0 and 1)
W_i	individual biomass (kg)

The consumed biomass can differ somewhat from the subtracted biomass, due to the introduced stochasticy. This effect can be removed by setting the discrete border of the prey species to zero or simulating high numbers of individuals.

3.3.3 Prey selection for fish

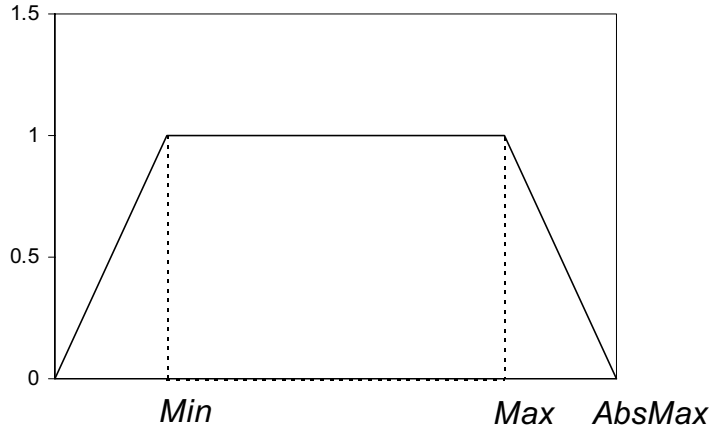
Consumption of predatory fish is modelled as type II functional response. The available prey biomass is calculated by multiplying the biomass with a specific selectivity. The selectivity is defined as follows:

$$Sel_{ij} = Hat(\frac{L_j}{L_i}) \, S_{ij}$$

in which:

Sel_{ij}	selectivity of predator j for prey i
Hat	'Hat' function (linearised unimodal function between 0 and 1) with the following parameters: Min = MinRatio, Max = MaxRatio, AbsMax = MaxRatio * 1.333 *SpecSelect*
L_j	Length of predator j (cm)
L_i	Length of prey i (cm)

Hat function



The 'hat' function is a linearised unimodal function. It has with five parts (Fig):

x < 0:	Hat = 0
0 < x < Min:	Hat = linear interpolation between (0, 0) and (Min, 1)
Min < x < Max:	Hat = 1
Max < x < AbsMax:	Hat = linear interpolation between (1, Max) and (AbsMax, 0)
x > AbsMax:	Hat = 0

3.3.4 Selection of zooplankton. mussels and zoobenthos

Zoobenthos, mussels and different groups of zooplankton are treated as uniform categories. Within these groups no explicit size distribution is considered.
The available food biomass of each of these groups is obtained by multiplying the specific selectivity (*ZoopSelect*) with the total biomass of each group.

3.3.5 Size dependent food switches

At any given moment only one food-type is consumed. The fishes can switch between the following categories of food:

1. fish: all fish species
2. zooplankton: Daphnia, other zooplankton
3. zoobenthos: all benthos species except mussels
4. mussels: Dreissena.

Switching between food items depends on predator size and food availability.

For instance, young breams feed on zooplankton. When they reach a size of ca. 15 cm, they switch to zoobenthos. But when there is much zooplankton they may feed on zooplankton for a while.

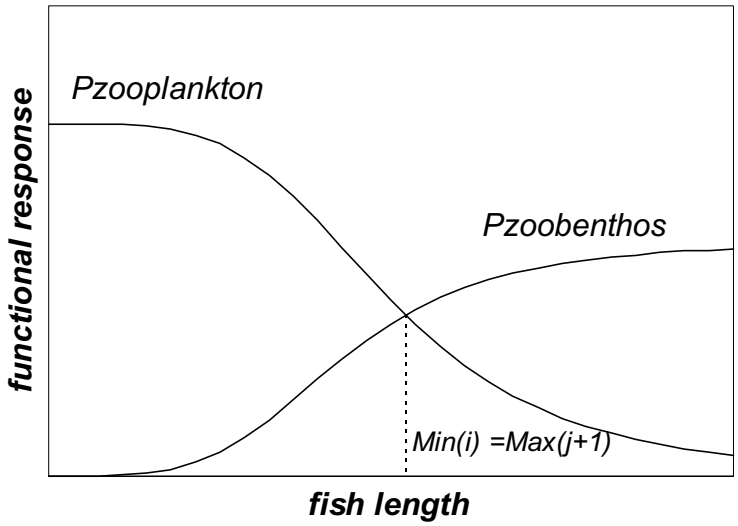


Fig. 2 Example of a food switch between zooplankton and zoobenthos (there is more zooplankton)

For each food item a range of predator length is defined. These ranges may overlap, because the switch is determined by food availability too. The functional response for each food item is multiplied with a Hill function with the predator length. The food item with the maximum result is chosen (Fig 2):

$$Fooditem = \underset{for\ i=1}{\overset{to\ n-1}{\text{maximum}}}\left(\frac{MaxLen_i^{pow}}{MaxLen_i^{pow} + L^{pow}}\, p_i\,, \frac{L^{pow}}{L^{pow} + MinLen_{i+1}^{pow}}\, p_{i+1}\right)$$

in which:

FoodItem	The food item that is chosen.
L	predator length (cm)
pow	power of the Hill function, typically a high value, e.g. 8 to get a sharp switch *FoodSwitch*.
$MinLen_i$	the approximate minimum length of the predator feeding on food item i (cm) *FoodSwitch*.
$MaxLen_i$	the approximate maximum length of the predator feeding on the food item i (cm) *FoodSwitch*.
p_i	the functional response of food item i (see also: functional response)

The changing of feeding strategy may result in a sub-optimal feeding behaviour. Therefore, the resulting functional response is the product of the functional response and the length function.

3.4 Reproduction

At a fixed day of the year (*BirthDay*) young fishes are introduced in the model. The number of young fishes per ha can either be fixed (if YOYSD = 0) or drawn at random from a statistical distribution (parameter: *YOYNumDistr*) with a mean of *YOYMean*, with standard deviation of *YOYSD*.

Alternatively, the data can be read from a text file (*YOYFileName*) with years and number of young fishes.

3.4.1 New fishes

The newly introduced fishes have an initial length (*YOYLen*). The initial weight is calculated using the length-weight relation.
The number of super-individuals (*nSuperInd*) can range between 1 and the total number of young fishes.

If the number of super-individuals is larger than one, individual variation can be introduced. We call the changing parameter the genetic parameter (*GenPar*). In the current version of Piscator the genetic parameter can be either the initial length of the fishes (*YOYLen*) or the average growth (*AvGrowth*). The coefficient of variation (CV) (=mean/SD) of the genetic parameter must be larger than 0 (*CVGenPar*).

The genetic parameters of the individuals are drawn from a normal distribution. However, the stochasticy of this repetitive drawing is neglected. The genetic parameters of each created super-individuals is distributed over the range of mean-SD*2.5 to mean+SD*2.5 with a fixed interval of (SD*5)/(nSuperind-1). The internal number of each super-individual equals the product of normal frequency distribution and the total number of new individuals.

See also: reproduction

3.4.2 Spawning

During the spawning period the fishes loose weight and may be more vulnerable to being caught, especially by fykes.

The start of the spawning period is predicted with a day-degree formula:

$$\sum_{t=1}^{n} (T_t - c) = d$$

in which

n	the number of days between March 1st and the spawning period.
T_t	the water temperature at day t (°C)
c	reference temperature (°C) *SpawnDayTemp*
d	the total heat sum defining the moment of spawning (°C) *SpawnDaySum*

The age of maturity is a parameter: *MaturityAge*.
The length of the spawning period is a parameter *SpawnDuration*. At the end of the spawning period the fishes loose a fraction of their weight *EggFraction*.

Note that there is no connection between spawning and reproduction.

The length after spawning can be adjusted by the following options (parameter *SpawnCalcType)*:
1. No change of the length (*SpawnCalcType = NoLengthChange*).
2. The length is adjusted in a way that the condition of the fishes remains unchanged (*SpawnCalcType = NoConditionChange*) (see also: length-weight relation). This is used to avoid unrealistic high starvation, and can be interpreted as the part of the length growth that was used for gonad production.
3. The fishes do not grow in length during the winter (*SpawnCalcType = NoWinterGrowth*).

3.5 Mortality

The following types of mortality are implemented:
1. mortality caused by predation
2. starvation mortality
3. senescence mortality
4. fixed mode mortality
5. background mortality

In all cases the mortality of a super-individual is executed by lowering the internal number. The internal number can either be discrete or real.
1. If the internal number is real the internal number is multiplied with the probability of mortality. The super-individual dies, if the internal number is lower than 0.5 *KillNumber*.
2. If the internal number is discrete, mortality is done by drawing from a binomial distribution with a sample size of the internal number (more convenient than repetitive drawing for each individual). If the internal number is 0 the super-individual dies.

The parameter *DiscrBord* defines the lowest discrete internal number. If the internal number exceeds 2,147,483,648 (the largest 32-bit integer) the internal number is always a real value (this happens for young smelt in Lake IJsselmeer!).

3.5.1 Background mortality

At each time step there is a fixed background mortality. It is simply modelled as a fixed parameter: *BackgrMort* (day^{-1})

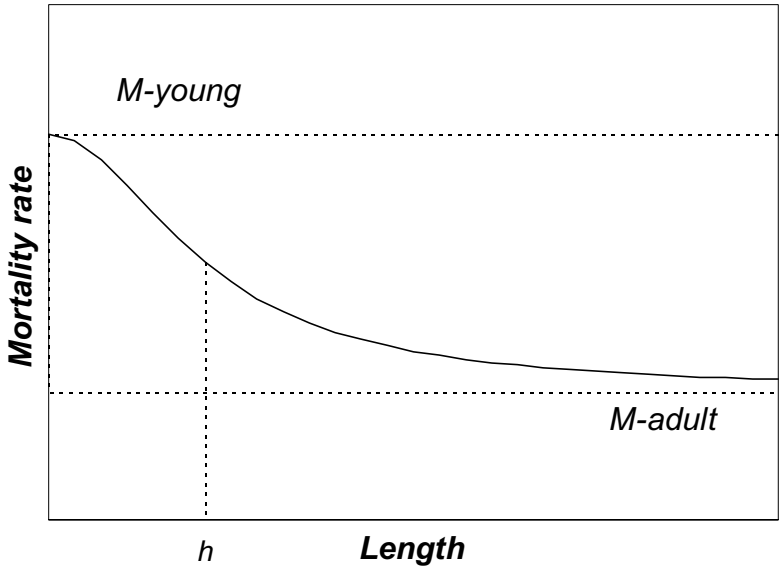


Fig. 3 Fixed mode mortality

3.5.2 Fixed mode mortality

If the mode of the individual is fixed mortality then all mortality is replaced by one background mortality, with the following temperature effect (Fig. 3):

$$m_f = \left[m_{f\,adult} + (m_{f\,young} - m_{f\,adult}) \frac{h_f^2}{h_f^2 + L^2} \right] Q_{10}^{\frac{T-T_r}{10}}$$

in which:

m_f	fixed mode mortality (day^{-1})
$m_{f\,adult}$	adult fixed mode mortality (day^{-1}) *FixedMort*
$m_{f\,young}$	young fixed mode mortality (day^{-1}) *FixedYoungMort*
h_f	parameter, the length where the mortality is halfway young and adult (cm) *HFixedMort*
L	length (cm)
Q_{10}	temperature effect (the same parameter as background mortality) *Q10Mort*
T	temperature (°C)
T_r	reference temperature (°C) *TRefMort*

Alternatively, a fixed mortality for each year class can be entered, along with a seasonal correction. The parameter *UserDefinedMort* is used to switch between both modes. In the user defined mode mortality is defined as follows:

$$m_f = m_{yclass}\, fact$$

m_f	fixed mode mortality (day^{-1})
m_{yclass}	mortality of the year class (day^{-1}) *YearClassMort*.
fact	seasonal correction (factor) *MonthlyMort*.

The fixed mode mortality is also calculated if the *mode* is *Dep. mortality*. Then, it is used for comparison only (by means of mortality tables or mortality graphs).

3.5.3 Starvation

If the weight of a super-individual is lower than about 0.7 (*CritFrac*) of the expected weight on basis of their length, starvation mortality takes place. The probability of dying is then typically 0.1 per day (*StarvMort*). See also: length-weight function.

3.5.4 Senescence

We have defined simply a 100% mortality if the maximum age for each fish (*MaxAge*). The mortality takes place the same day that new fishes are introduced (*BirthDay*)

4. Fishery

Three types of fishery are defined:
1. fykes
2. gill-nets
3. seines

Gill-nets, fykes and seines are selective for size and species. In the model nets can be considered as consuming, but not-growing individuals. The consumption is dependent on the number of net units and the density of vulnerable fish. The functional response is considered as a functional response type I (linear).

4.1 Gill-nets

The food-consumption of gill-nets is modelled as a functional response type 1:

$$c_t = e\ N\ TotFB_t\ s_t$$

in which:

c_t	catches of gill-nets at day t (kg ha^{-1})
e	efficiency per net (linear functional response) *NetEff*
$TotFB_t$	total vulnerable fish biomass: a function of mesh size, length of prey, specific selectivity (kg)
s_t	relative seasonal effort (a monthly number between 0 and 1) *SeasonEff*
N	Number of nets (ha^{-1}) *NNets*

The total vulnerable fish biomass is calculated analogous to the consumption of predatory fishes (see: total available prey biomass), but with a different selectivity function. Gill-nets are very length selective. We assume a normal distribution of the optimal size. Fishes are vulnerable to being caught in gill-nets in a narrow range of length selectivity:

$$Sel_{ij} = S_j\ e^{-0.5\ \frac{(\frac{L_i\ LM_j}{Mesh} - \mu)^2}{\sigma^2}}$$

in which:

Sel_{ij}	selectivity of individual i of species j *SpecSelect*.
L_i	length of prey i (cm)
LM_j	length-mesh conversion of species j (dependent of the shape of the fish, as a rule of thumb 1/9) *SpecSelect*.
S_j	specific selectivity (a value between 0 and 1)
Mesh	mesh size of the gill-net (cm) *NetSize* (there can be created several gill nets (*NMeshes*) with different sizes and number *NumNets*)

m	mean of the normal distribution, fixed value = 1
s	standard deviation of the normal distribution, fixed value = 0.1

Optionally, gill nets can be distributed unequally over habitats.

4.2 Fykes

The food-consumption of fykes is modelled as a functional response type 1:

$$c_t = N\ TotFB_t\ s_t$$

in which:

c_t	catches of fykes at day t (kg ha^{-1})
$TotFB_t$	total vulnerable fish biomass: a function of length of prey and specific selectivity (kg)
s_t	relative seasonal effort (a monthly number between 0 and 1) *SeasonEff*.
N	Number of nets (ha^{-1}) *NNets*

The efficiency of fykes for some species increases strongly in the spawning period. Therefore, two parameters are defined for each species: efficiency in the period

The total vulnerable fish biomass is calculated analogous to the consumption of predatory fishes (see: total available prey biomass), but with a different selectivity function. The selectivity is a combination of specific selectivity and length selectivity:

$$Sel_{ij} = Hat(\frac{L_{pred}}{L_{ij}})\ S_j$$

in which:

Sel_{ij}	efficiency/selectivity of individual i of prey species j
Hat	'Hat' function (linearised unimodal function between 0 and 1) with the following parameters: Min = MinLength, Max = MaxLength, AbsMax = MaxLength *SpecSelect*
L_{ij}	Length of individual i of prey species j (cm)
S_{ij}	Specific selectivity of predator j for the species of prey i (ranges between 0 and 1) *SpecSelect*

Optionally, fykes can be unequally distributed over habitats.

4.3 Seine nets

The food-consumption of seines is modelled as a functional response type I:

$$c_t = N\ TotFB_t$$

in which:

c_t	catches of seines at day t (kg ha^{-1})
$TotFB_t$	total vulnerable fish biomass: a function of mesh size, length of prey, specific selectivity (kg ha^{-1})
N	Intensity of fishery (area fished per lake area) *NumNets*

The fishery takes place at discrete events defined by the parameter *SeasonEff*.

The total vulnerable fish biomass is calculated analogous to the consumption of other nets and of predatory fishes (see: total available prey biomass), but with a different selectivity function. We assume that all fishes larger than a threshold length are caught with equal probability. For each species an efficiency can be defined:

$$if\ L_i\ LM_j > Mesh\ \ then\ \ Sel_{ij} = S_j\ \ else\ \ Sel_{ij} = 0$$

in which:

Sel_{ij}	selectivity for prey i of species j
L_i	length of prey i (cm)
LM_j	length-mesh conversion of species j (dependent of the shape of the fish, as a rule of thumb 1/4) *SpecSelect*
S_j	specific selectivity (a value between 0 and 1) *SpecSelect*
Mesh	mesh size of the seine (cm) *NetSize* (there can be created several seine nets (*NMeshes*) with different sizes and number *NumNets*)

Optionally, seine nets can be unequally distributed over habitats.

5. Food species

5.1 Piscivorous birds

Consumption by cormorants, grebes, smews and mergansers. The effect of birds is comparable to nets because the preference for specific species and size does not vary. This is because the size of the birds hardly changes. Its feeding response is modelled in an all or none way. If the food concentration drops below a limit, the birds move to another area for feeding. The concentrations and consumption per day are monthly constraints or entered as *data file*)

If $TotFB_t > FB_{crit}\ Area$ then

$$B = N_t C_t\ Area$$

else If $TotFB_t > FB_{min}\ Area$ then

$$B = N_t C_t\ Area \frac{TotFB_t - FB_{crit}}{FB_{\min} - FB_{crit}}$$

else

$$B = 0$$

in which:

B	consumption of fish by birds (kg ha^{-1})
C_t	maximal food intake per individual (kg) (entered as monthly value *DailyCons*)
N_t	number of birds (entered as means per month *MeanNum*)
FB_{crit}	critical prey biomass, if the food biomass is below this value part of the birds migrate *CritPreyBiomass*.
FB_{min}	absolute critical prey biomass *MinCritPreyBiomass*.
$TotFB_t$	total food biomass: sum of all fishes smaller than maximum prey length *MaxLen* and larger than the minimum prey length *MinLen*.

Optionally, these species can be unequally distributed over habitats.

5.2 Zooplankton

The concentration of zooplankton is fixed (a monthly value, see parameter *ConcArray* or *DataFileName*). These species can be unequally distributed over habitats.

As an alternative a mini model is implemented (switch parameter *UserDefinedC*). The model uses two compartments, a refuge for fish predation and open water. The parameter *Refuge* determines the ration of the lake area used as refuge of zooplankton. Fish predation takes place outside the refuge, in the open water, exclusively. The diffusion between the compartments is modelled in a similar way as Scheffer and De Boer (1995) used in their model of phytoplankton.

$$\frac{dZ_1}{dt} = (r\,u\frac{A}{h_A + A} - m_b)Z_1\,Q_{10}^{\frac{T-20}{10}} + \frac{D}{f}(Z_r - Z_1)$$

$$\frac{dZ_r}{dt} = (r\,u\frac{A}{h_A + A} - m_b)Z_r\,Q_{10}^{\frac{T-20}{10}} - \frac{D}{1-f}(Z_1 - Z_r)$$

in which:

Z_1	biomass zooplankton in open water (kg ha^{-1})
Z_r	biomass zooplankton in refuge (kg ha^{-1})
t	time (day)
A	biomass of phytoplankton (kg ha^{-1})
h_A	half-saturation of the functional response (kg ha^{-1}) *HFuncResp*
Q_{10}	temperature effect *Q10_K*
r	maximal consumption of zooplankton (d^{-1}) *MaxConsump*
u	utilisation efficiency (fraction) *UtilEff*
f	fraction of lake area used as refuge for zooplankton (fraction) *Refuge*
D	diffusion between both compartments (d^{-1}) *Diffusion*
T	water temperature (^{o}C)
m_b	background mortality (d^{-1}) *MBackgr*

In case that the refuge fraction is (close to) zero, the model is simplified as follows:

$$\frac{dZ_1}{dt} = (r\,u\frac{A}{h_A + A} - m_b)Z_1\,Q_{10}^{\frac{T-20}{10}}$$

$$Z_r = NAN \text{ (= not available number)}$$

The biomass of phytoplankton is modelled as a logistic growing food (the same way as zoobenthos and mussels).

In the fixed growth mode the functional response is replaced by a fixed functional response (*FuncResp*). In the dependent mode the consumption term is subtracted from the algal biomass.

5.3 Zoobenthos, mussels and phytoplankton

The concentrations of zoobenthos and mussels are fixed monthly values (parameter *ConcArray* or *DataFileName*). These species can be unequally distributed over habitats.

As an alternative mini models are implemented (switch parameter *UserDefinedC*). The model uses two compartments, a refuge for fish predation and open water. The parameter *Refuge* determines the ration of the lake area used as refuge. Fish or zooplankton predation takes place outside the refuge, in the open water, exclusively. The diffusion between the compartments is modelled in a similar way as Scheffer and De Boer (1995) used in their model of phytoplankton.

$$\frac{dA_1}{dt} = r(1 - \frac{A_1}{k_t})A_1 + \frac{D}{f}(A_r - A_1)$$

$$\frac{dA_r}{dt} = r(1 - \frac{A_r}{k_t})A_r - \frac{D}{1-f}(A_1 - A_r)$$

in which:

A_1	biomass logistic growing species in open water (kg ha^{-1})
A_r	biomass logistic growing species in refuge (kg ha^{-1})
t	time (day)
r_t	growth rate, corrected for temperature: r $Q_{10r}^{(T-20)/10}$, T = temperature °C, r = growth rate at 20°C, parameter: *R*, Q_{10r} = constant *Q10_r*.
k_t	carrying capacity, corrected for temperature: k $Q_{10k}^{(T-20)/10}$, T = temperature °C, k = carrying capacity at 20°C, parameter: *K*, Q_{10k} = constant *Q10_k*.
f	fraction of lake area used as refuge for zooplankton (fraction) *Refuge*
D	diffusion between both compartments (d^{-1}) *Diffusion*

6. Research group

The Piscator team consists of:

Eddy Lammens	biologist, project leader (RIZA)
Egbert van Nes	program development (Wageningen Agricultural University)
Marten Scheffer	mathematical biologist (Wageningen Agricultural University)

A panel of experts has evaluated the project yearly. Members of the group:

Tom Buijse (DFL-LI)
Mennobart van Eerden (DFL-LI),
Wim van Densen (Wageningen Agricultural University)
Peter Mous (LUW, RIVO)
Wolf Mooij (NIOO)
Willem Dekker (RIVO)

Contact address:

Eddy Lammens
RIZA
PO Box 17
8200 AA Lelystad
tel. +31 320 298411
E-mail: E.Lammens@RIZA.RWS.MinVenW.NL

Egbert van Nes

WAU
PO Box 8080
6700 DD Wageningen
E-mail: Egbert.vanNes@aqec.wkao.wau.nl

7. References

De Hoop, B.J., P.M. Herman, H. Scholten, & K. Soetaert, 1992. SENECA 1.5. A simulation environment for ecological application. Manual. Netherlands Institute of Ecology 180 pp.

Klepper, O., 1989. A model of carbon flows in relation to macrobenthic food supply in the Oosterschelde estuary (S.W. Netherlands).

Klepper, O. & E.M. Hendrix, 1994a. A comparison of algorithms for global characterization of confidence regions for nonlinear models. Environmental Toxicology and Chemistry. 13: 1887-1899.

Klepper, O. & E.M. Hendrix, 1994b. A method for robust calibration of ecological models under different types of uncertainty. Ecol.Model. 74: 161-182.

Price, W.L., 1979. A Controlled Random Search procedure for global optimization. The Computer Journal. 20: 367-370.

Scheffer, M., J.M. Baveco, D.L. DeAngelis, E.H. Lammens, & B. Shuter, 1995. Stunted growth and stepwise die-off in animal cohorts. Amer.Naturalist. 145: 376-388.

8. Appendixes

8.1 Dutch translation of some terms

bream	brasem
cormorant	aalscholver
eel	aal, paling
flounder	bot
fykes	fuiken
gill-nets	staande netten
goosander	grote zaagbek
grebe (great crested)	fuut
merganser (red-breasted)	middelste zaagbek
perch	baars
pikeperch	snoekbaars
roach	blankvoorn
ruffe	pos
smelt	spiering
smew	nonnetje
seine	zegen (net)

8.2 Statistical distributions

The program supports several statistical probability distributions. Here follows a very short description of the used distributions for continuous variables. The program can generate (if necessary) random numbers that have one of these. In the program these distributions are all characterized by their mean (= expected value) and their SD (= square root of the variance), or their minimum (mostly defined as 5% quantile) and maximum (95% quantile).

Normal distribution
The normal distribution is one of the most important statistical distibutions used in probability. It is useful for describing a variety of random processes such as students' test scores or the heights of trees.
A normal distribution is fully described with just two parameters: its *mean* (mu) and *standard deviation* (sigma). The distribution is a continuous, bell-shaped distribution which is symmetric about its mean and can take on values from negative infinity to positive infinity.
P = 1 / (sigma * sqrt(2*pi)) * exp(-sqr(x - mu) / (2 * sqr(sigma)));

Lognormal distribution

The lognormal distribution is a statistical distribution, which is defined as follows: a random variable is lognormally distributed if the natural log of the variable is *normally* distributed.
P = 1 / (x * sigma * sqrt(2*pi)) * exp(-sqr(ln(x) - mu) / (2 * sqr(sigma)))

Uniform distribution
The uniform distribution is simply a random number drawn within a certain range. The function is fully defined by the minimum and the maximum.

x < min:	P = 0
min < x < max:	P = (x - min) / (max-min)
x>max:	P = 0

LogUniform distribution
A stochastic variable is LogUniform distributed if the natural log of the variable is uniformly distributed.

x < min:	P = 0
min < x < max:	P = 1 / x (ln(max)-ln(min))
x>max:	P = 0

Exponential distribution
The exponential distribution is a very skewed distribution: it is an exponentially decreasing function, that is defined by a minimum and a standard deviation.

x < min:	P = 0
x > min:	P = 1 / beta * exp(-(x - alfa) / beta)

Logistic distribution
The logistic distribution is another bell shaped function. The same probability function is used in logistic regression.
P = exp(-(x - alfa) / beta)/ (beta * sqr(exp(-(x - alfa) / beta)))

Weibull distribution
The shape of Weibull distribution can range from a decreasing function to a bell-shaped function. This function is particularily useful for variables that must be positive.
P = afa * (x / beta)^alfa)* exp(-(x / beta)^alfa))

8.3 Format of data files

Data files are used in the model for several purposes. A flexible delimited format is used for all data files. It has the following restrictions:

The rows of the data file must be delimited by carriage returns (Enter).

The columns of the data file must be delimited by a comma, semicolon, tab or space(s). The first line of the data file is used to determine which delimiter should be used.

If the number of columns is not constant for all rows, the data file can not be read.

If a value is not numerical, it is treated as missing value. An exception is if the first row doesn't contain digits. Then, the first line is treated as column titles.

The first column may contain dates in a standard format: YYYYMMDD.

Any empty line in the file is discarded.