

AmP projects: setting up

Starrlight Augustine

starrlight@tecnico.ulisboa.pt

University of Lisboa/ Vrije
Universiteit

Dina Lika

lika@uoc.gr

University of Crete

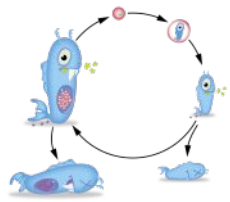


MARETEC
MARINE, ENVIRONMENT
AND TECHNOLOGY CENTER
TÉCNICO LISBOA



University of
Crete

School: 4-13 June 2023
Baton Rouge, Louisiana, United States
deb2023.sciencesconf.org

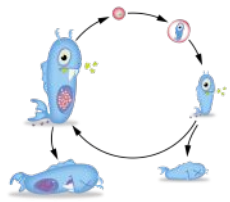


Overview

- Set up individual objectives and plan for the “own project” time : 14.5 Hrs
- Final day: 3 hours total of AmP presentations.
Groups of 1 till three people. 7 min/person - or 10 min/ group

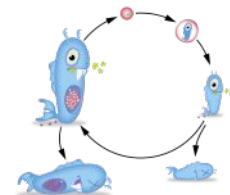
Main topic of this lecture:

Auxiliary theory



Learning Objectives

- Know the difference between core DEB theory assumptions and auxiliary theory assumptions
- Know how to navigate the AmP database and associated resources to find user-defined predictions for you type of data



Know your data labels

debportal.debtheory.org/docs/AmPestimation.html

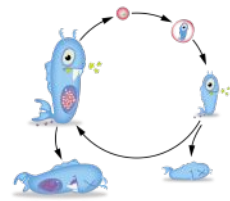
DEB portal

AmP estimation

es the methodology that is used in the [AmP project](#) for the estimation of parameters of DEB models from and getting started, technical aspects, i.e. code-specifications and how to use it to arrive at parameter estimation of estimation results, accuracy of parameter estimates. An introduction to modelling and statistics is present [Basic methods for Theoretical Biology](#). The described procedure is coded in the DEBtool software ([GitHub](#)) and follows the [DEB notation](#).

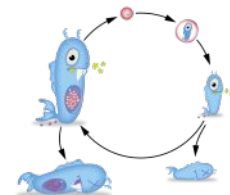
tion is the procedure to use data to arrive at parameter estimates by minimizing some loss function that measures the difference between data and model predictions, known as the **estimation criterion**. A loss function is a function of model predictions and weight coefficients, so the problem is to find the combination of parameter values that minimizes the loss function, starting from one or more initial guesses. Several methods (algorithms) can be used for this, but the choice should be independent of the algorithm. A complicating factor is that the loss function typically has several minima, meaning that neighbouring parameter values have higher values of the loss function. The global minimum is the global minimum, which is usually at the parameter estimate that we are looking for. But there might be other local minima, and the global and local minima might be small, and a local minimum might be at parameter values that have no physiological interpretation. Typical for parameter estimation in an AmP context is that we not have a single data set, but several to many, which are all used simultaneously to estimate all parameters in a single estimation step.

First change equations and assumptions in your notes



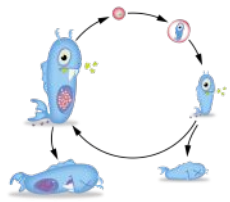
Only after that can change them in the code

When modelling with the PC you need to distinguish between an error coding a quantity and a model term that need to be changed because assumptions have been updated.



Some suggestions

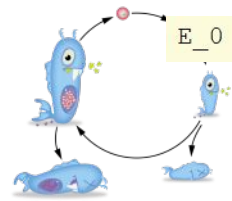
- Have zotero standalone with DEB library
- Have handy the papers with the most useful appendixes (or main papers) with the equations you will be using.
- Always have DEB notation handy. Follow the nomenclature in the equations but also how the code relates to the symbols.
- Write units in comments next to quantities in code



Data finding takes the most time

- Stay on top of references and units for data.
- Make notes of important details to help modelling and interpretation
- Do not transform data in the data file

Work with standardized nomenclature

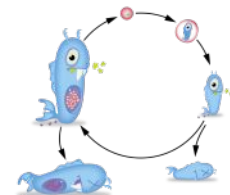


DEB nomenclature

Standardized link between symbols and **code**

	E_0	E_0
	a_b, a_p	aT_b
	a_m	
units ←	L_b, L_p	
	L_∞	
	W_w^b	
	W_w^p	Ww_p
	W_w^∞	
	N_∞	

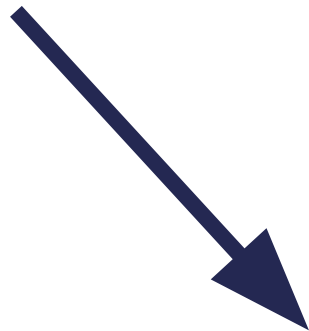
Work with your own equation notes



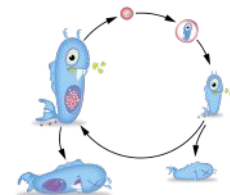
$$L(t) = L_{\infty} - (L_{\infty} - L_b) \exp(-t\dot{r}_B) \quad \text{or} \quad t(L) = \frac{1}{\dot{r}_B} \ln \frac{L_{\infty} - L_b}{L_{\infty} - L}$$

$$\dot{r}_B = \frac{1}{3/\dot{k}_M + 3fL_m/\dot{v}} = \frac{\dot{k}_M/3}{1 + f/g}$$

$$L_{\infty} = fL_m - L_T$$

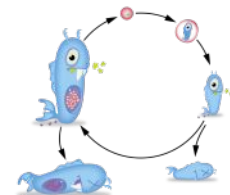


```
% time-length
rT_B = kT_M/ 3/ (1 + f_tL/ g); L_i = L_m * f_tL; L_b = L_m * get_lb([g k v_Hb], f_tL);
L = L_i - (L_i - L_b) * exp( - rT_B * tL_f(:,1)); % cm, structural length at time
ELw_f = L/ del_M; % cm, shell length
```



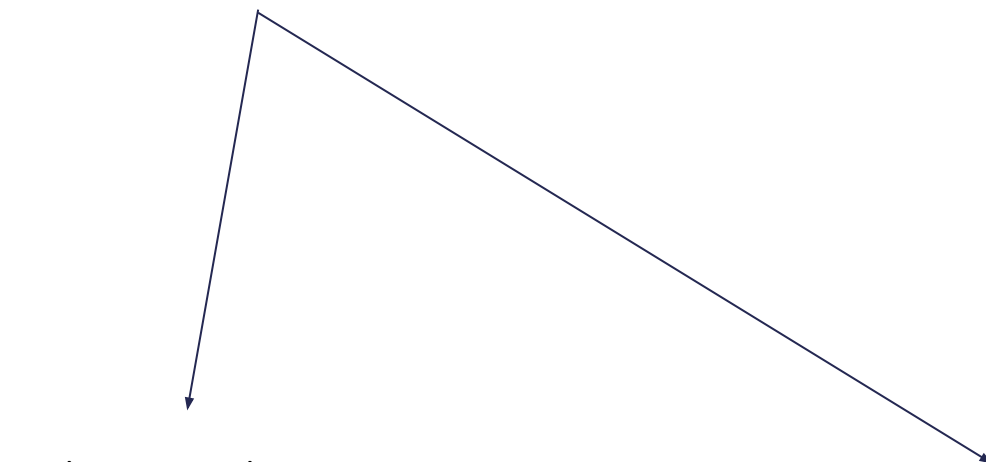
Calculating Quantities with DEBtool:

Symbol	Units	Interpretation	
E_0	J	initial energy in egg	<code>initial_scaled_reserve</code>
a_b, a_p	d	age at birth, puberty	<code>get_tj</code>
a_m	d	age at death	<code>get_tm_mod</code>
L_b, L_p	cm	structural length at birth, puberty	<code>get_tj</code>
L_∞	cm	ultimate volumetric length	$s_{\mathcal{M}} f L_m$
W_w^b	g	wet weight at birth	$L_b^3(1 + f\omega)$
W_w^p	g	wet weight puberty	$L_p^3(1 + f\omega)$
W_w^∞	g	ultimate wet weight	$L_\infty^3(1 + f\omega)$
N_∞	#	life time reproductive output	<code>cum_reprod</code> (std-DEB) or <code>cum_reprod_j</code> (abj-DEB)



Finding predictions

Know your labels !!! debportal.debtheory.org/docs/Uni-variate_data.html

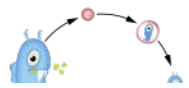


Look in species list

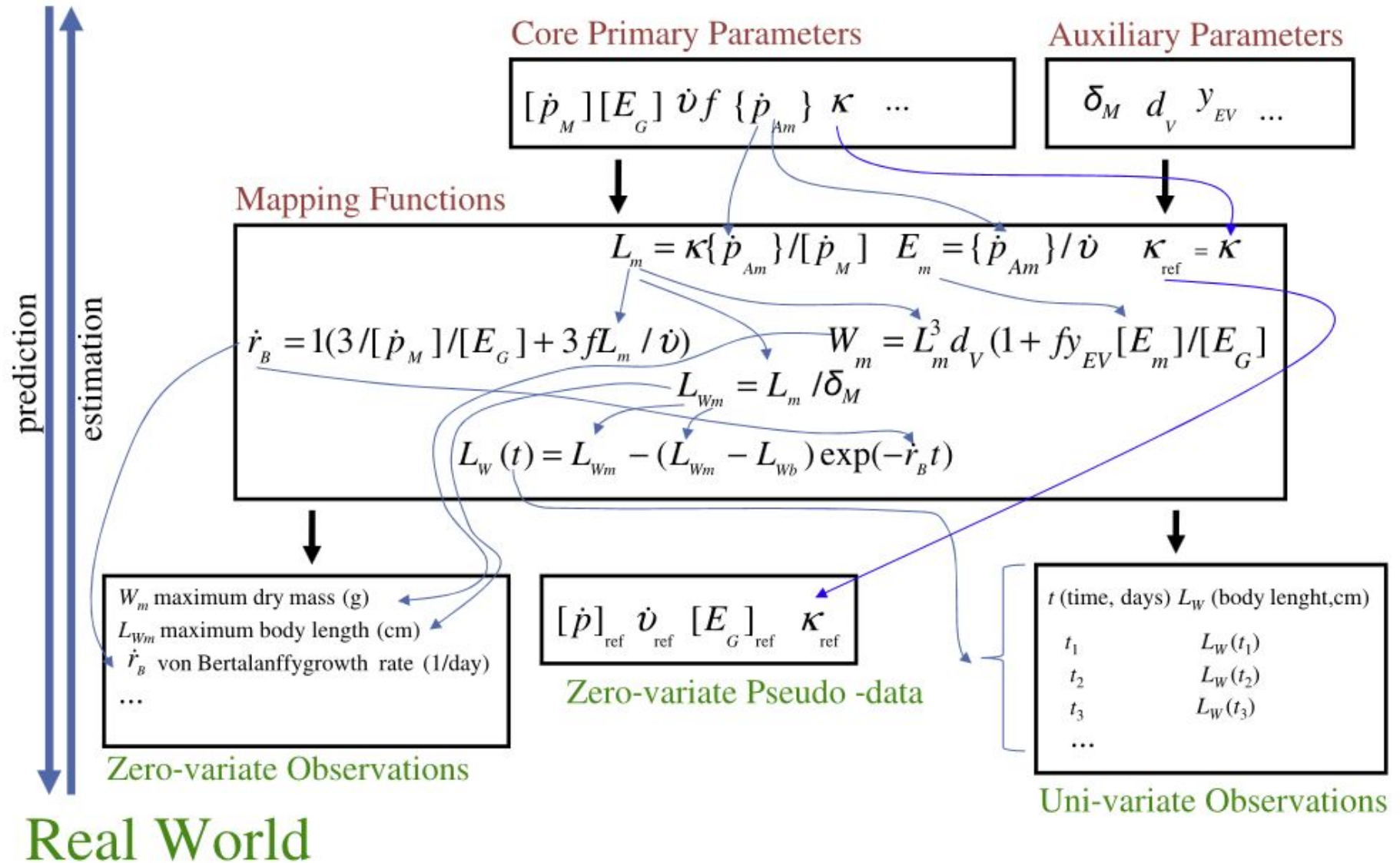
Use AmPtool functions amptool.debtheory.org/docs/

```
select_data({'t-Le', 'Wwb'}, 'std')
```

```
[species, nm] = select_predict('f = spline1(t, tf)'); select_predict(nm, 'males')
```



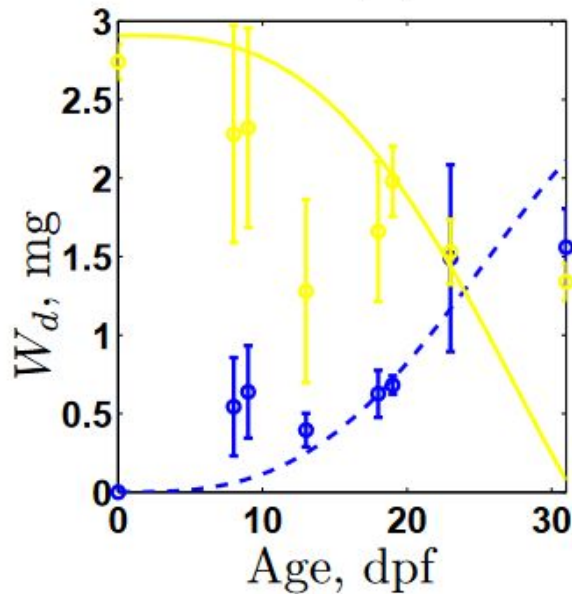
Abstract World





Examples of “maps” :

Embryo yolk
and body:



Abstract World

Model

prediction
estimation

Organism data

Real World

State Variables

V, E, E_R

Core Primary Parameters

$\dot{v}, [\dot{p}_M], \{\dot{p}_{Am}\}, [E_G], \kappa, \dots$

Auxiliary Parameters

$\delta_M, \mu_E, w_E, d_V, \dots$

Mapping Functions

$$e_b = \frac{E_b}{V_b}$$

$$W_Y = E \frac{w_E}{\mu_E} - e_b m_{Em} w_E [M_V] V$$

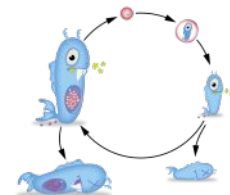
$$W_{emb} = (w_E - e_b m_{Em} w_E) [M_V] V$$

time (d)	Yolk dry mass (g)
t_1	$W_Y(t_1)$
t_2	$W_Y(t_2)$
t_3	$W_Y(t_3)$
...	

Uni-variate Observations

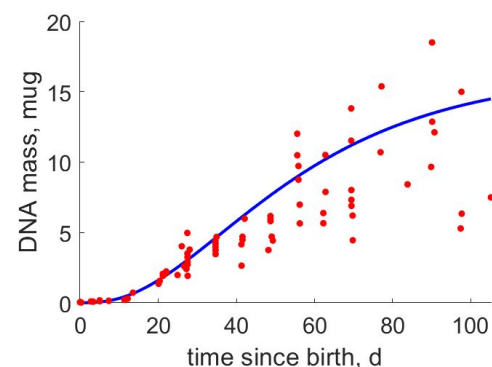
time (d)	Embryo dry mass (g)
t_1	$W_{emb}(t_1)$
t_2	$W_{emb}(t_2)$
t_3	$W_{emb}(t_3)$
...	

Uni-variate Observations



Examples of “maps”

DNA and RNA weights



Abstract World

Model

prediction

estimation

Organism data

Real World

State Variables

V, E, E_R

Core Primary Parameters

$\dot{v}, [\dot{p}_M], \{\dot{p}_{Am}\}, [E_G], \kappa, \dots$

Auxiliary Parameters

$Y_{DNA}, Y_{RNA}, \mu_E, w_E, d_V, \dots$

Mapping Functions

$$[E_m] = \frac{\{\dot{p}_{Am}\}}{\dot{v}}$$

$$w = \frac{[E_m]w_E}{d_V\mu_E}$$

$$e = \frac{E}{V}$$

$$M_{DNA} = Y_{DNA}d_VV$$

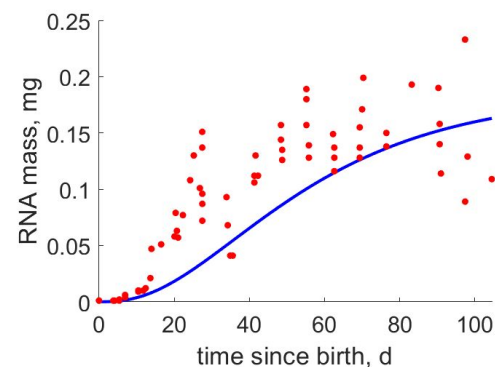
$$M_{RNA} = Y_{RNA}d_VwVe$$

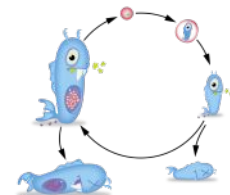
time (d)	DNA mass (g)
t_1	$M_{DNA}(t_1)$
t_2	$M_{DNA}(t_2)$
t_3	$M_{DNA}(t_3)$
...	

Uni-variate Observations

time (d)	RNA mass (g)
t_1	$W_{emb}(t_1)$
t_2	$W_{emb}(t_2)$
t_3	$W_{emb}(t_3)$
...	

Uni-variate Observations

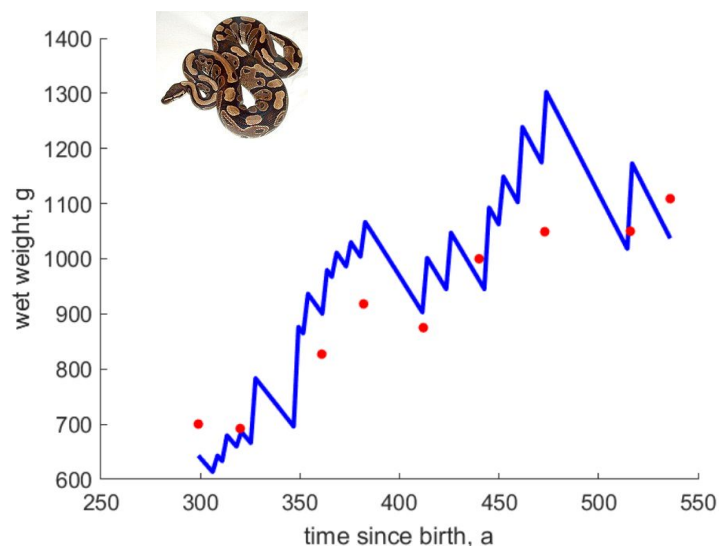




Examples of “maps”:

Python regius eats mice

User defined function in the predict file to include weight of gut content:

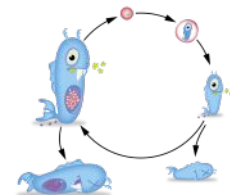


```
function dXeL = dXeL(t,XeL, vT, pT_Am, g, L_m, L_T)
% routine for predict_Python_regius
% digestion at max rate

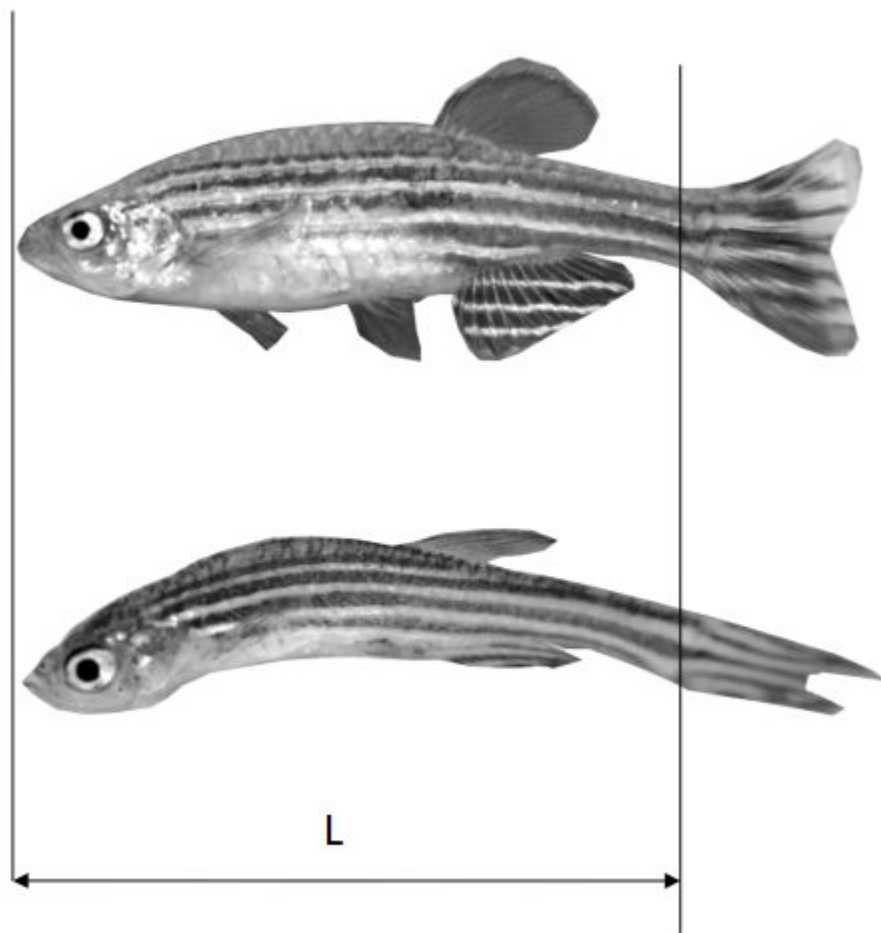
%unpack state variables
X = max(0, XeL(1)); e = XeL(2); L = XeL(3);

dX = - (X > 0) * pT_Am * L^2;
de = - dX/ L^3 - e * vT/ L;
r = vT * (e/ L - (1 + L_T/ L)/ L_m)/ (e + g);
dL = L * r/ 3;

dXeL = [dX; de; dL];
end
```

Length as quantifier for structure



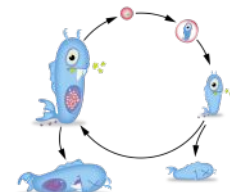
Structure



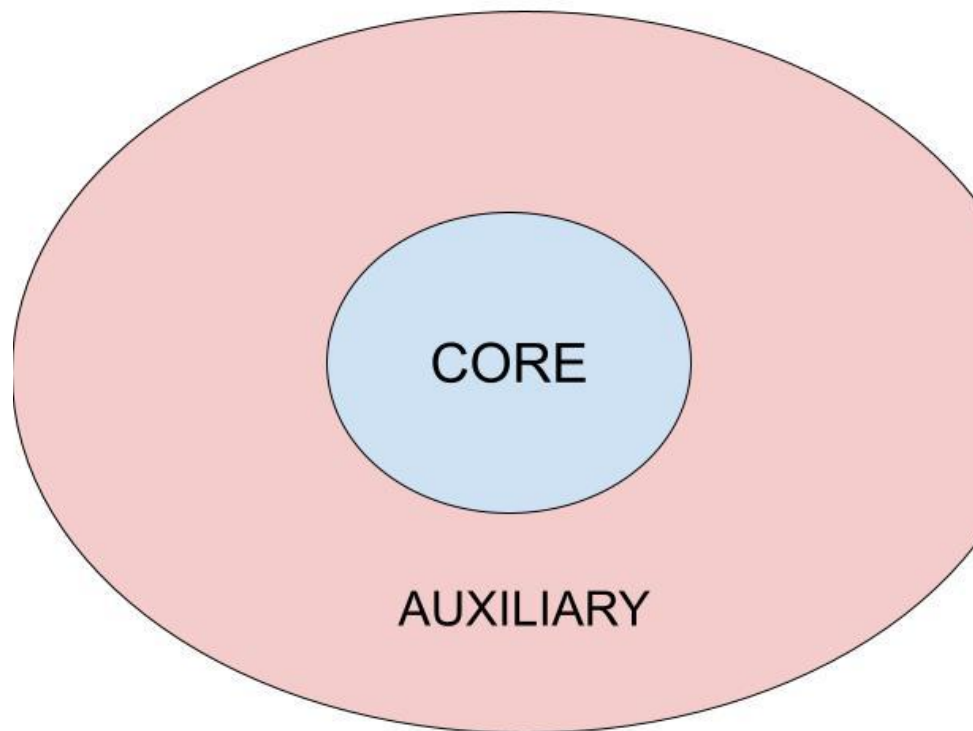
Structure +
Reserve

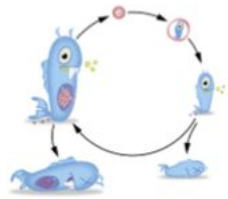


Different reserve density :
amount of reserve per unit
structure



Assumptions....

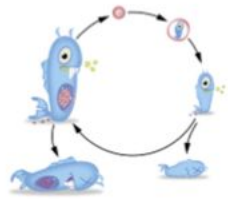




mydata-file

Set the data

- **metadata** - information about: the species, eco-codes, the types of data, the author, etc
- **data** - value, units, labels, references, temperature (for time and rates)
- **pseudo data** - were set automatically, can be modified



Data

(real)data

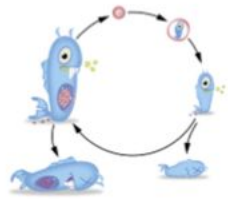
Empirical observations of physiological processes

- zero-variate (single measurements)
- uni-variate (lists of values for an independent and an associated dependent variable)
- Read [here](#) for more types of data

pseudo-data

Prior knowledge of a selection of parameter values

Fill possible gaps in information in the real data



mydata-file

```
%% set weights for all real data
```

```
weights = setweights(data, []);
```

```
%% set pseudodata and respective weights
```

```
[data, units, label, weights] = addpseudodata(data, units,  
label, weights);
```

```
%% pack auxData and txtData for output
```

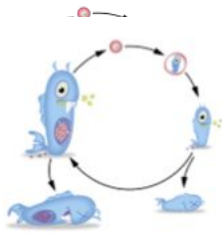
```
auxData.temp = temp;
```

```
txtData.units = units;
```

```
txtData.label = label;
```

```
txtData.bibkey = bibkey;
```

```
txtData.comment = comment;
```



pars_init-file

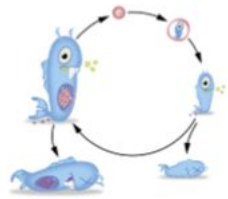
Rename the file and the function: **pars_init_Squalus_acanthias**

Choose the model

Set the initial parameter values for estimation

including info for units, labels and if it is a free parameter (i.e., parameter that is allowed to change during the estimation procedure)

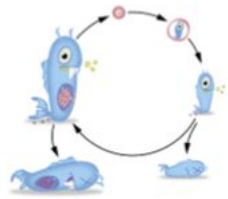
- Primary
- Auxiliary
- Environmental
- Chemical



Structures in Matlab

- A **Structure** is a collection of data representing a single idea or "object"
- Inside a structure there is a **list of fields** each being a variable name for some sub-piece of data
- Different type of data for each "field"
- Syntax

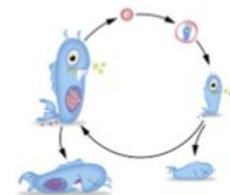
`VariableName . Field`



Structure example

A single variable (named `metaData`) which contains many **fields**. Each field has its own **name** and its own **type**.

```
metaData.phylum      = 'Chordata';  
metaData.class         = 'Chondrichthyes';  
metaData.order         = 'Squaliformes';  
metaData.family        = 'Squalidae';  
metaData.species       = 'Squalus_acanthias';  
metaData.species_en    = 'Spurdog';
```



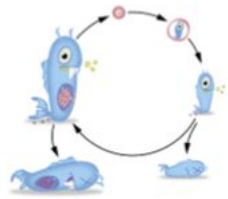
Structure example

Many variables (named `data`, `units`, `label`, `bibkey`, `temp`)
which contain many **fields** (e.g. zero-variate `data`, `ab`, `ap`).

Variables give different information for a given field.

```
data.ab = 640.5  
units.ab = 'd'  
label.ab = 'age at birth';  
bibkey.ab = 'JoneGeen1977';  
temp.ab = T_C + 9
```

```
data.ap = 2190  
units.ap = 'd';  
label.ap = 'age at puberty';  
bibkey.ap = 'Avsa2001'  
temp.ap = T_C + 9;
```

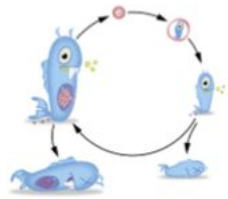



Structures in Matlab

Use function `struct`

```
s = struct(field1,value1,field2,value2)
```

Creates a structure array with multiple fields



Pack/unpack variables

`vars_pull` : pack & unpack variables into & from structures, according to the syntax and inputs.

Example : unpack variables from data

```
vars_pull(data)
```

creates or overwrites variables `ab` and `ap` in the calling function with the contents of the corresponding named fields