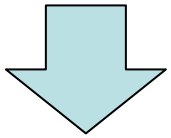
Using R for mathematical modelling (the environment).

Karline Soetaert

NETHERLANDS INSTITUTE OF ECOLOGY

Natural systems are very complex

❑ Scientists want to understand this complexity and make quantitative predictions



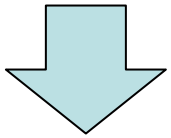
❑Model = simplifications of the complex natural environment

❑Test model to data

❑Quantification of unmeasured processes

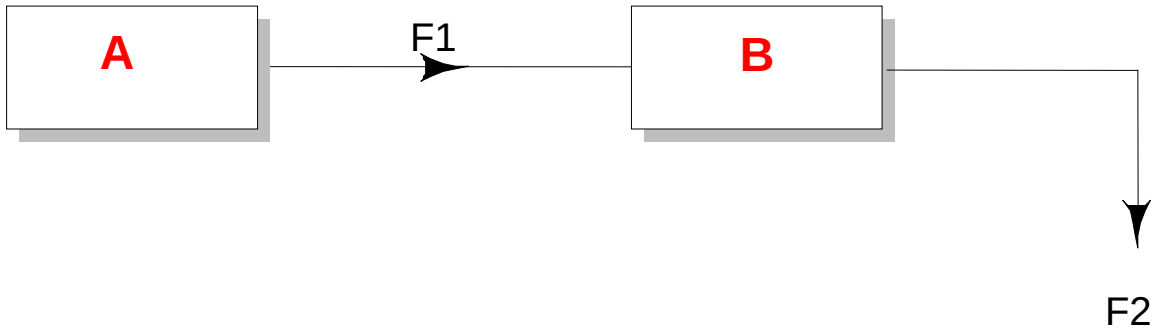
❑Budgetting, interpolation in time/space

❑….



❑Prediction of future behavior

- Mathematical model based on mass balance conservation



$$\frac{dA}{dt} = -F1$$

$$\frac{dB}{dt} = F1 - F2$$

$\Rightarrow$ Differential equations

- Use R to solve mathematical mass balance models

- Three different types of models/solutions – three main packages

  - Integration (deSolve)
  - Steady-state solution (rootSolve)
  - Least-squares solutions (limSolve)

  What was available + what is new

- Two examples

  - HIV model (dynamic / steady-state)
  - Deep-water coral food web

Example 1: hiv dynamics

Large interest in viral infection

- Human disease
- Marine animals, algae, bacteria are affected

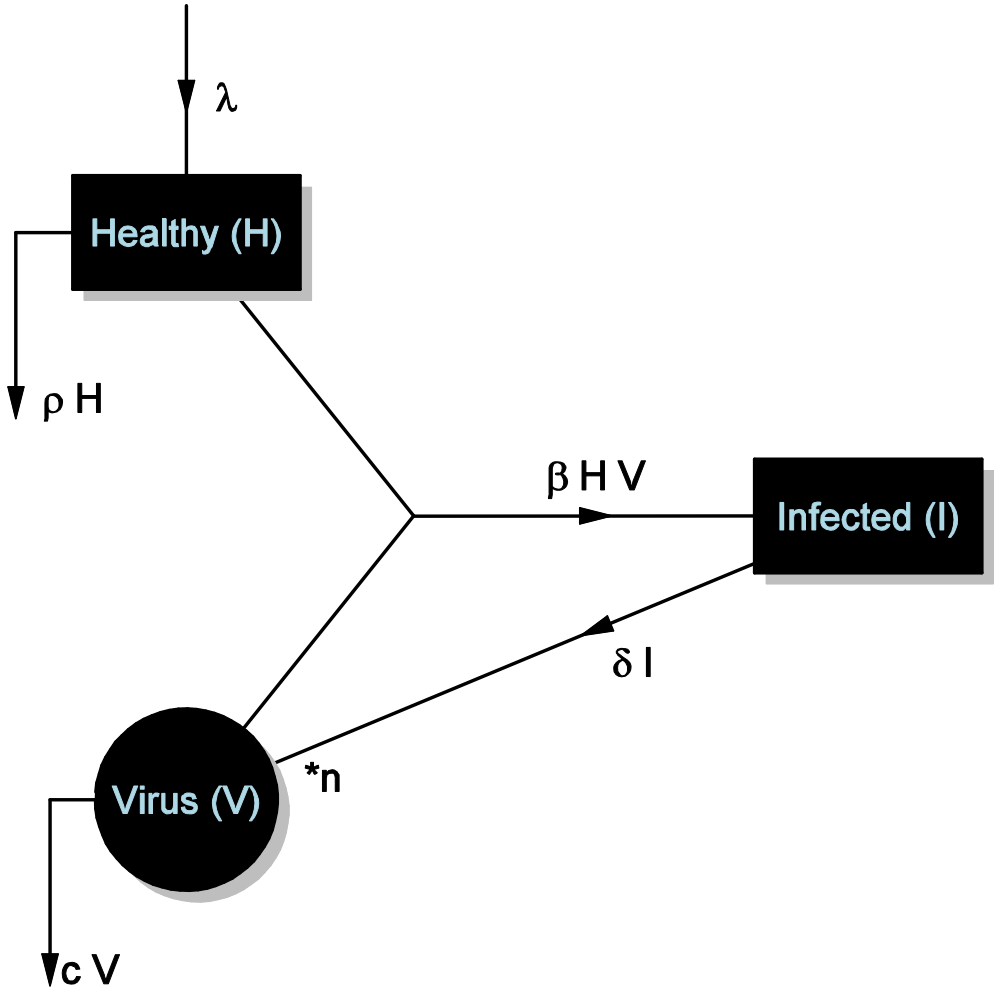
$\Rightarrow$Important role in biogeochemical cycles

$$\frac{dH}{dt} = \lambda - \rho \cdot H - \beta \cdot H \cdot V$$

$$\frac{dI}{dt} = \beta \cdot H \cdot V - \delta \cdot I$$

$$\frac{dV}{dt} = n \cdot \delta \cdot I - c \cdot V - \beta \cdot H \cdot V$$

t = time
n,c,… = parameter
H,I,V = state variable

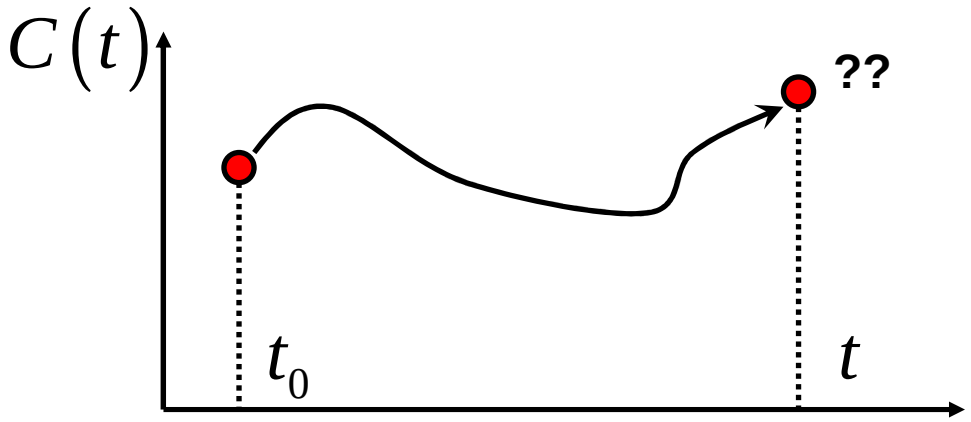


Model formulation:

Derivative $\frac{dC}{dt} = f(\Theta, C, t, u)$

Initial condition $C_{t=0} = C0$

Model solution:
integration $C(t)$ **For t > t_0**



Previously on CRAN: odesolve (Setzer 2001)

- Nice interface
- Two integration routines:
  - *RungeKutta*, not meant to be used
  - *lsoda*, good for small, simple models
- Models implemented in R or compiled code DLL (fast)

BUT:

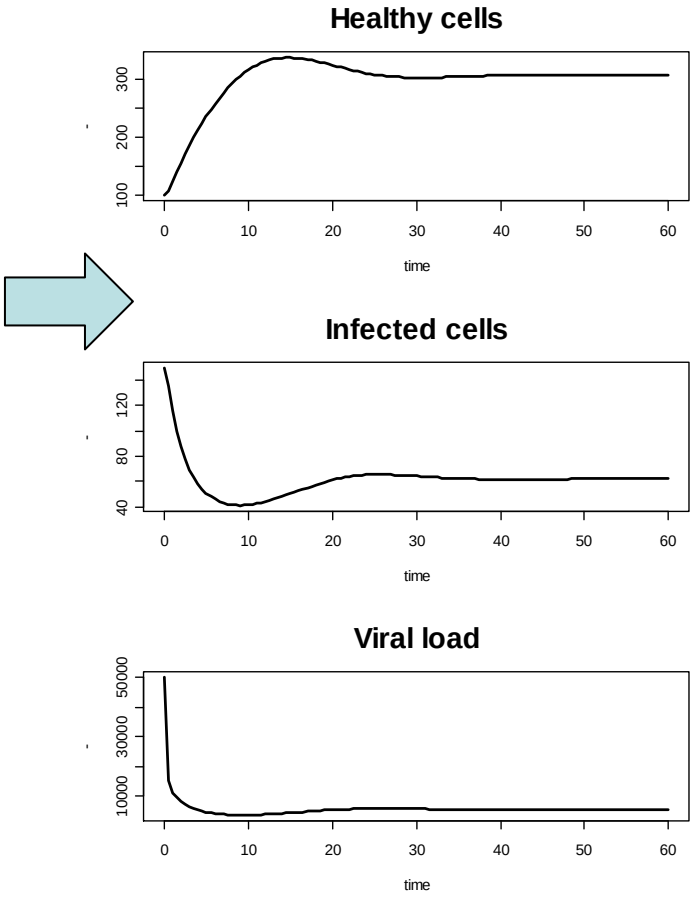
- Only simplest Ordinary Differential Equations (ODE)
- not flexible,
- not suited for large problems

Now on CRAN:

- **deSolve** (Soetaert, Petzoldt, Setzer)
  - Initial value problems
  - > 10 integration routines
  - Simple and complex ODE
  - Differential algebraic equations (DAE)
  - Partial differential equations: (PDE)
    - 1-D, 2-D, 3-D problems
  - Flexible; Sparse, banded, full Jacobian
  - Medium-sized to large problems (up to 80000 state variables)
- **bvpSolve** (Soetaert)
  - Boundary value problems
  - 2 solution methods

The HIV/AIDS model in R

```
hiv<- function( time, y, pars) {
    with (as.list(c(pars,y)), {

      dH <- lam -rho*H - bet*H*V
      dI   <- bet*H*V -delt*I
      dV <-  n*delt*I - c*V - bet*H*V

      return( list(c(dH, dI, dV)) )
    })
  }

y <- c(H= 100,  I = 150,  V = 50000)

times <- 0:60

pars<- c(bet=0.00002,rho=0.15,delt=0.55,c=5.5,lam=80,n=900)

out <- ode(y=y, parms=pars, times=times, func=hiv)

 plot(out,"time"], out,"H"])
```



Problem: dynamic models require many data:

$$\frac{dC}{dt} = \sum f_i(C, \Theta, t, u, ...) - \sum f_j(C, \Theta, t, u, ...)$$

- Knowledge of initial values
- Time-variable forcing functions (external data, u)

=> Not always available

Solution:

$$0 = \sum f_i(C, \Theta, ...) - \sum f_j(C, \Theta, ...)$$

- Assume **steady-state**
  - =>Systems of **nonlinear equations**

- Calculate **stability** properties

Previously on CRAN:

- *uniroot* solves for one root of one nonlinear equation within interval

We need:

- Find all roots within one interval
- Functions to estimate gradient matrices, Jacobians (stability)
- Solve roots of n nonlinear equations (steady-state analysis)

Now on CRAN:

- **rootSolve** (Soetaert)
  - *uniroot.all* , *jacobian*: stability analysis
  - *multiroot* : roots of general nonlinear functions (Newton-Raphson)
  - *steady, steady.1D, steady.2D, steady.3D, runsteady* : steady-state solvers
    - Fully compatible with integration routines from **deSolve**
    - Suited for large problems (~100 000 equations)
    - Sparse, banded, full Jacobian

```
STD <- runsteady(y=y, func=hiv, parms=pars)

eigen( jacobian.full (y=STD$y, func=hiv, parms=pars) )$values
```

Problem:

- ❑mechanistic nonlinear models have many parameters (θ):

  ⇒Many are unknown
  ⇒need to be fitted to data
  ⇒Data not always available

$$\frac{dC}{dt} = \sum f_i(C, \Theta, t, ...) - \sum f_j(C, \Theta, t, ...)$$

- ❑nonlinear equations may not be known

Solution:

- ❑Avoid nonlinear equations
- ❑No parameters
- ❑The sources and sinks ($f_{i->j}$) are the unknowns

  ⇒Linear model

$$\frac{dC_j}{dt} = \underbrace{\sum f_{i \to j}}_{Sources} - \underbrace{\sum f_{j \to k}}_{Sinks}$$

$$\frac{d\mathbf{C}}{dt} = \mathbf{A} \cdot \mathbf{x}$$

Example 2: Deep-water coral food webs

Corals are commonly found at ~ 800-1000 m water depth.

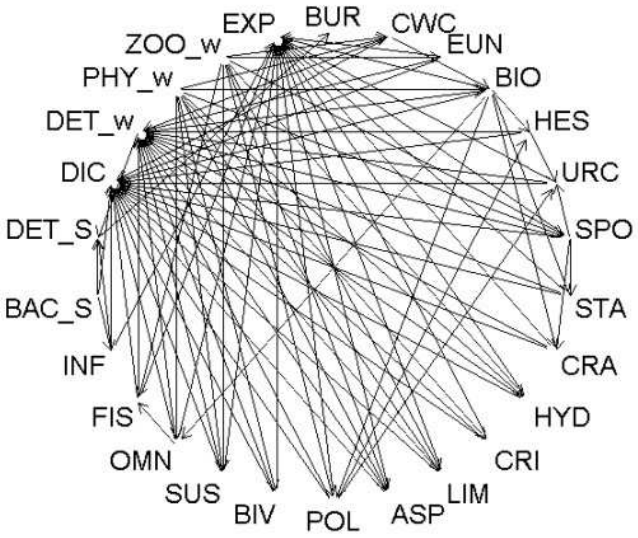
A large number of animals are living in the coral reefs

It is very expensive to do research there

⇒Data are very fragmentary

⇒Who is eating who? How much do they eat?

⇒A model is needed to see the global picture

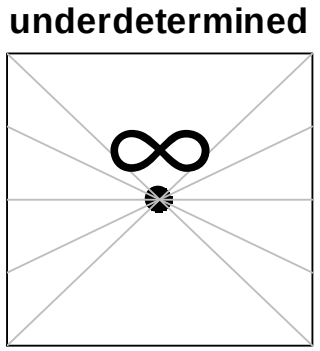


Problem:

number of equations <<< number of unknowns (under determined)

Coral food web: 51 equations ~ 140 unknowns

⇒There is no unique solution
(~ fitting a straight line through one point)



Solution 1. Add data from other sources to equalities
=>achieve overdeterminacy (1 solution)

Solution 2. Data from other sources as "inequalities"

equality equation: $\mathbf{Ex = f}$

(in situ data, mass balance)

inequality equation: $\mathbf{Gx \geq h}$

(literature data, physiological constraints,..)

linear functions — numerical data

$$\begin{pmatrix} a & -b & 0 \\ \vdots & & \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \geq \begin{pmatrix} C1 \\ \vdots \end{pmatrix}$$

food web flows

» the matrix equations are solved for the vector with food web flows

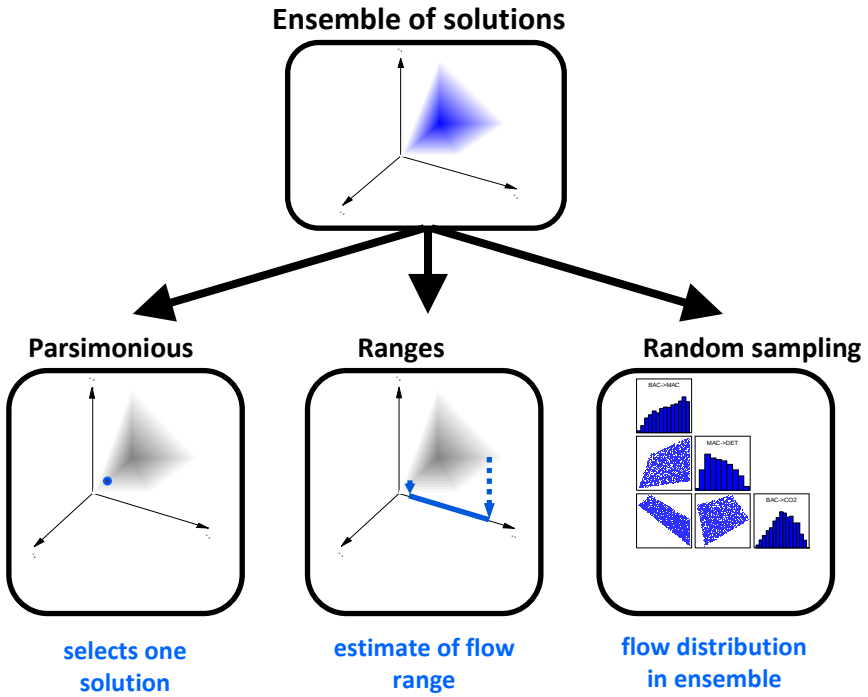
Dealing with the underdeterminacy:

$$\mathbf{Ex} = \mathbf{f}$$

$$\mathbf{Gx} \geq \mathbf{h}$$

Coral : Solution is a 140-dimensional SPACE!
⇒Within this space, every point equally likely

⇒3 different ways of solving:



$$\min \left\| \mathbf{Ax} \approx \mathbf{b} \right\| \qquad \min(x)$$

$$\min(\sum x^2) \qquad \max(x)$$

Previously on CRAN

- *solve.qp,* (**quadprog)**: quadratic programming

$$\min \left\| \mathbf{Ax} \approx \mathbf{b} \right\|, \mathbf{Ex} = \mathbf{f}, \mathbf{Gx} \geq \mathbf{h}$$

- *lp* , (**lpSolve)**: linear programming

$$\min(\sum a_i x_i), \mathbf{Ex} = \mathbf{f}, \mathbf{Gx} \geq \mathbf{h}$$

But:

- solve.qp tends to fail for some problems
- lp requires x to be positive (linear programming)
- lp and solve.qp are not compatible
- No monte carlo sampling of underdetermined systems
- Implementing large matrices: error-prone

Now on CRAN:

limSolve (Soetaert, van Oevelen, van den Meersche)

- ❑least squares,
- ❑linear programming,
- ❑least distance programming
- ❑*xranges, xsample:* range estimation and random sampling

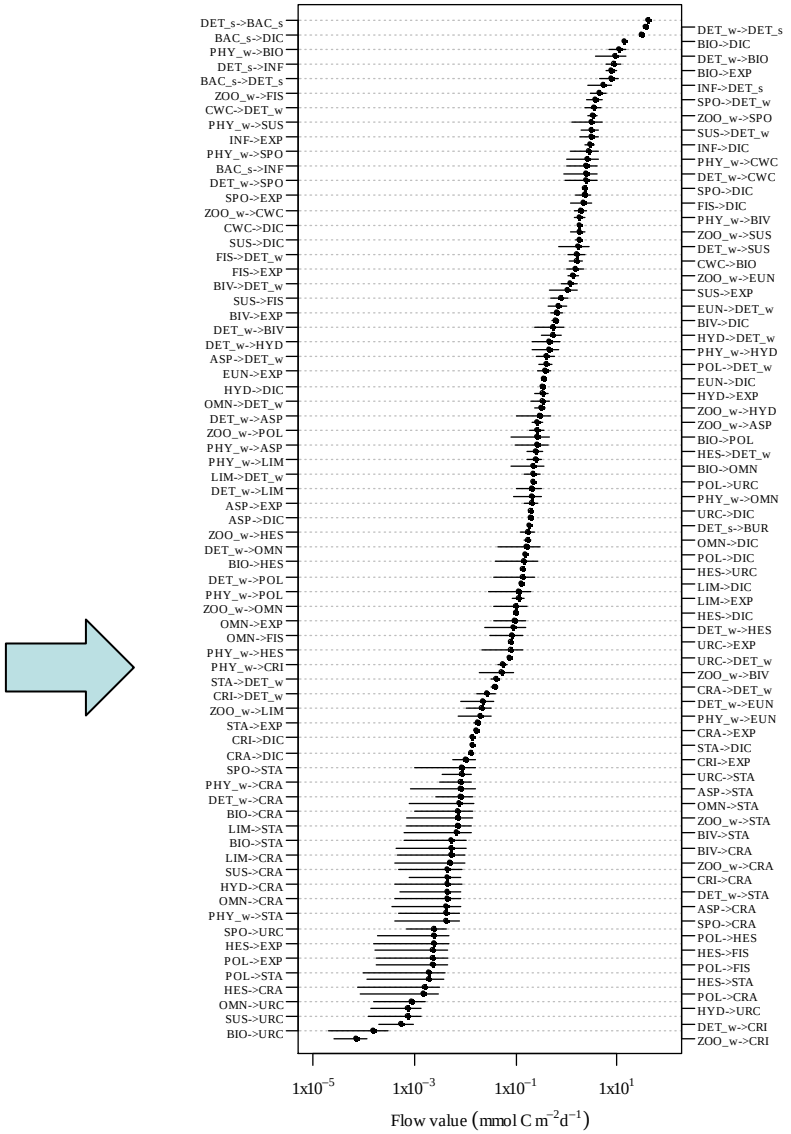
LIM (Soetaert, van Oevelen)

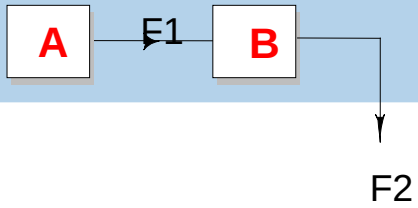
- ❑Models are specified in text files

```
Flowto(CO2) = 100

coral -> CO2 = [0.2,0.4] * Flowto(coral)
…
```

“coral.input”

```
require(LIM)
coral.lim        <- Setup(“coral.input")

Parsimonious <-Ldei(coral.lim)
Ranges  <- Xranges(coral.lim)
Xs          <- Xsample(coral.lim, iter=10000)

Plotranges(order(colMeans(Xs)),…)
…
```





Linear

limSolve

LIM

$$\frac{dC_j}{dt} = \overbrace{\sum flow_{i\to j}}^{input} - \overbrace{\sum flow_{j\to k}}^{output}$$

Steady-state nonlinear

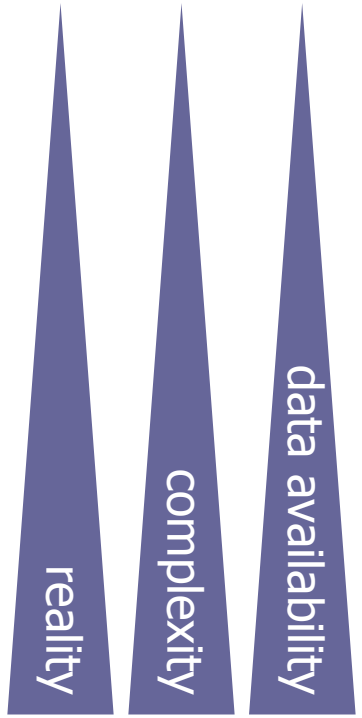
rootSolve

$$0 = \sum f_i(C,\Theta,\ldots) - \sum f_j(C,\Theta,\ldots)$$

Dynamic

deSolve

$$\frac{dC}{dt} = \sum f_i(C,\Theta,t,u,\ldots) - \sum f_j(C,\Theta,t,u,\ldots)$$



- Before 2006: Fortran, Excel, Powerpoint, Sigmaplot, own software
- End 2005. First acquaintance with R
- End 2006. Decision to use R for our scientific programming / graphics

  $\Rightarrow$ Implement functions not yet available

- Three years later...

  ⇒Basic solution methods available
  ⇒5 Solver packages (deSolve, rootSolve, bvpSolve, limSolve, LIM)
  ⇒Specific model applications

- Reactive transport models, (ReacTran)
  ⇒rivers, estuaries, lakes, sediments

- Toxicology, (ToxWebs)
  ⇒ toxic substances in marine organisms

- Ecological network analysis (NetIndices)

- ….

THANK YOU

....

Soetaert K. and P.M.J. Herman, 2009. A practical guide to ecological modelling – using R as a simulation platform. Springer, 372 pp

Soetaert, K., van Oevelen, D., 2009. Modeling food web interactions in benthic deep-sea ecosystems: a practical guide. Oceanography (22) 1: 130-145.