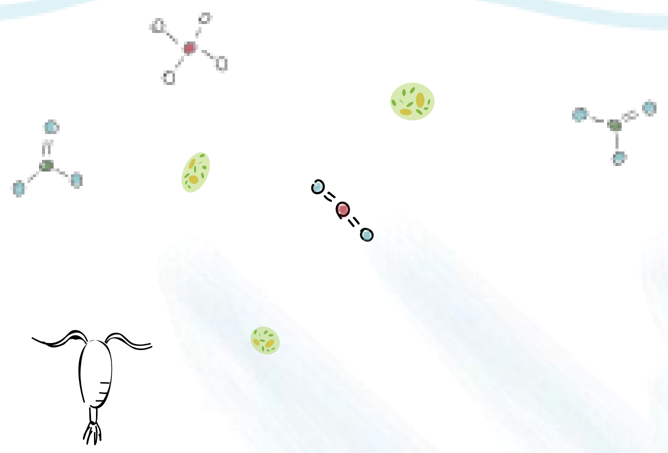


Reactive Transport in the Hydrosphere

Department of Earth Sciences, Faculty of Geosciences, Utrecht University

Lecturers: Lubos Polerecky and Karline Soetaert

Illustrations, narration and video editing: Renee Hageman
Additional contributions: Dries Bonte, University Ghent
Audio effects: mixkit.co

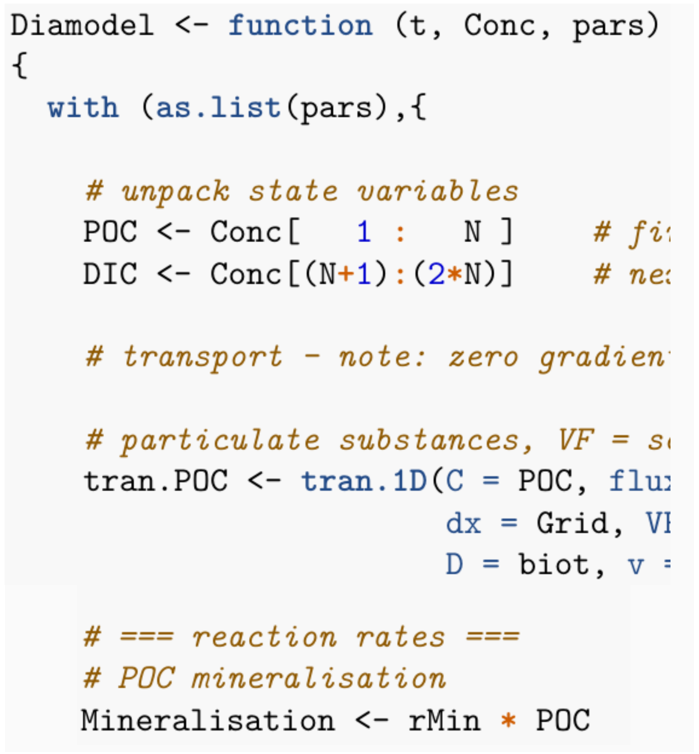
Universiteit Utrecht

GHENT UNIVERSITY

Solving reaction-transport equation using the R-package **`ReacTran`**

Ingredients of a reaction-transport model

1. Conceptual diagram
2. Mass balances
3. Transport flux
4. Reaction-transport equation
5. Effects of porosity
6. Multi-component models
7. Boundary conditions



```
Diamodel <- function (t, Conc, pars)
{
  with (as.list(pars),{

    # unpack state variables
    POC <- Conc[   1 :    N ]      # fi
    DIC <- Conc[(N+1):(2*N)]      # ne

    # transport - note: zero gradien

    # particulate substances, VF = s
    tran.POC <- tran.1D(C = POC, flu
                        dx = Grid, V
                        D = biot, v

    # === reaction rates ===
    # POC mineralisation
    Mineralisation <- rMin * POC
```

Universiteit Utrecht

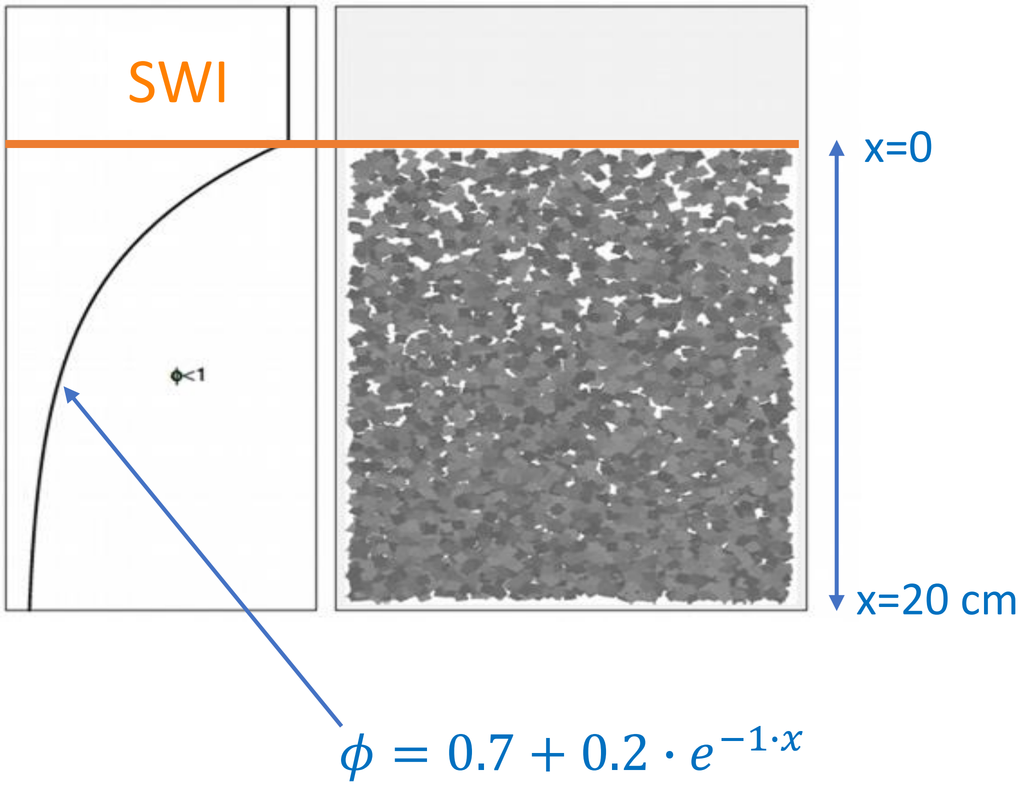
GHENT UNIVERSITY

“Skeleton” of a ReacTran R-code

	`ReacTran` functions:
1. Define spatial **grid** and model **parameters**	`setup.grid.1D` `setup.prop.1D`
2. Define and initialize **state variables**	`state <- c(. . .)`
3. Define **model function** (transport and reaction terms, time-derivatives)	`tran.1D` `dA.dt <- ...`
4. Calculate and display model **results**	`steady.1D` `ode.1D` `plot` `image`
5. Check **mass balances** and make **budget**	

Step 1: Problem definition

- Particulate organic carbon **mineralization** in sediments: **POC** → **DIC**



$$\phi = 0.7 + 0.2 \cdot e^{-1 \cdot x}$$

State variables:

POC	$(mol\ C\ m_s^{-3})$
DIC	$(mol\ C\ m_L^{-3})$

Domain:

Space: 0 – 20 cm

Time: days

Step 2: Rate expressions and mass balances

- POC **mineralization** is first-order process, and the only reaction

$$Mineralisation = r_{Min} \cdot [POC] \qquad r_{Min} = 0.01\ d^{-1}$$

$$\uparrow$$

$$(\boldsymbol{mol\ C\ m_s^{-3}\ d^{-1}})$$

- Mass balance equations:

$$\frac{d[POC]}{dt} = -Mineralisation + tran_{POC} \quad ✓ \quad (\boldsymbol{mol\ C\ m_s^{-3}\ d^{-1}})$$

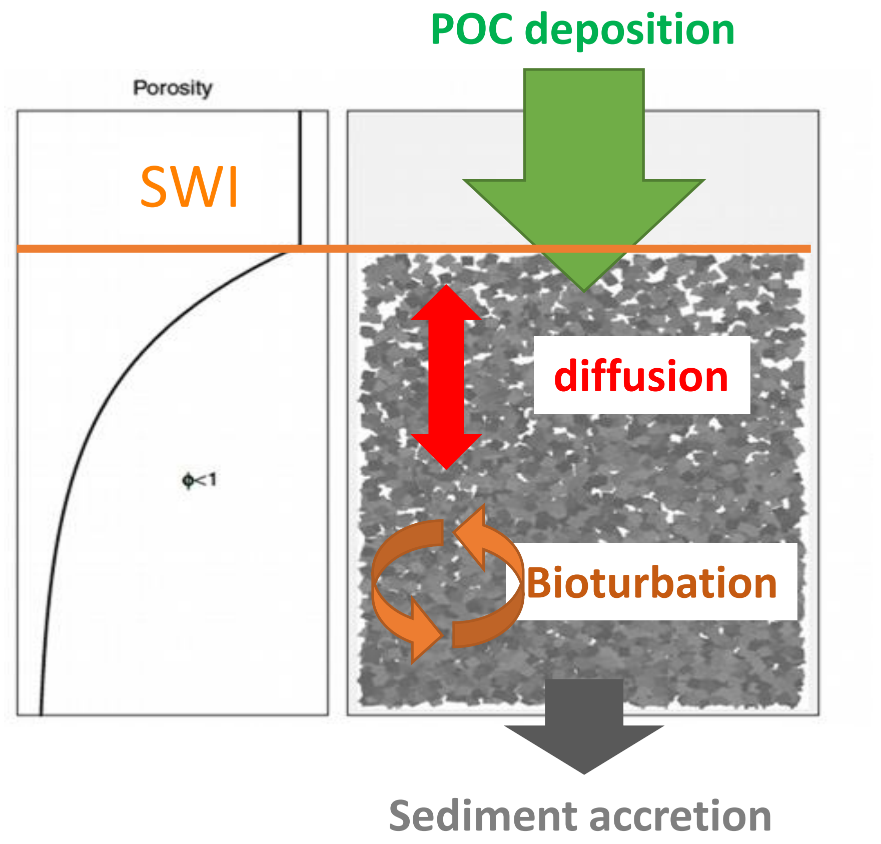
$$\frac{d[DIC]}{dt} = Mineralisation \cdot \frac{1-\phi}{\phi} + tran_{DIC} \quad ✓ \quad (\boldsymbol{mol\ C\ m_L^{-3}\ d^{-1}})$$

Universiteit Utrecht

GHENT UNIVERSITY

Step 3: Environmental settings

- **Transport processes** and **boundary conditions**



$DEPO_{POC} = 100\ mmol\ C\ cm^{-2}\ d^{-1}$

$DIC_{SWI} = 2000\ nmol\ C\ cm_L^{-3}$

$D_{mol}(DIC) = 0.98\ \times 10^{-9}\ m^2\ s^{-1}$
$\approx 0.85\ cm^2\ d^{-1}$

$D_{bio} = 5\ cm^2\ yr^{-1} \approx 0.0137\ cm^2\ d^{-1}$

$v = 0.005\ cm\ d^{-1}\ \approx 1.8\ cm\ yr^{-1}$

$temperature = 20\ °C, salinity = 35$

Universiteit Utrecht

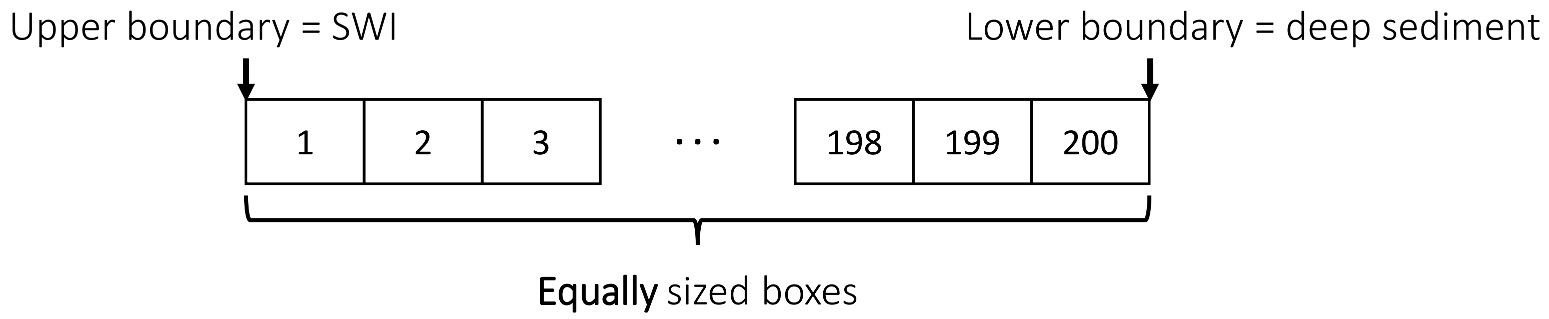
GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

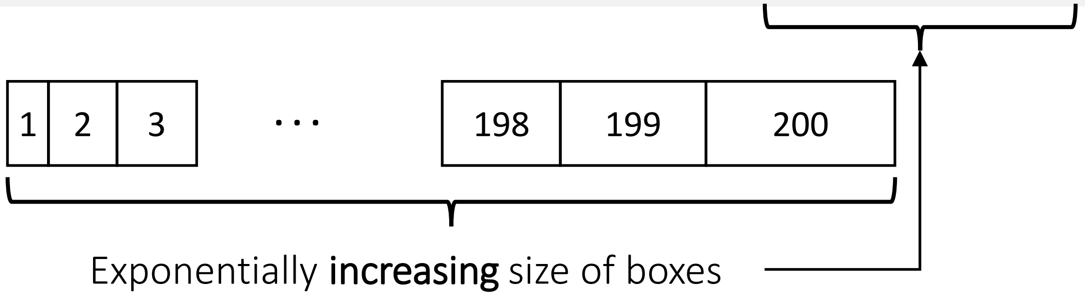
4.1 Divide the spatial domain into a **grid**

```
Length <- 20
N      <- 200

Grid   <- setup.grid.1D(L = Length, N = N)
```



```
Grid   <- setup.grid.1D(L = Length, N = N, dx.1 = 0.05)
```



Step 4: Implementation in R (ReacTran)

4.2 Define parameters on the **grid**

Porosity and solid volume fraction:

```
porFun.L  <- function(x, por.SWI, por.deep, porcoef)
  return( por.deep + (por.SWI-por.deep)*exp(-x*porcoef) )

porFun.S <- function(x, por.SWI, por.deep, porcoef)
  return( 1-porFun.L(x, por.SWI, por.deep, porcoef) )
```

Evaluation on the grid:

```
porLiquid <- setup.prop.1D(func = porFun.L, grid = Grid,
  por.SWI = 0.9, por.deep = 0.7, porcoef = 1)

porSolid <- setup.prop.1D(func = porFun.S, grid = Grid,
  por.SWI = 0.9, por.deep = 0.7, porcoef = 1)
```

Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.2 Define parameters on the **grid**

Diffusion coefficient at grid interfaces

```
diffHCO3  <- diffcoeff(S=35, t=20)$HCO3 * 3600*24*1e4
```

salinity temperature species

Evaluation on the grid:

porosity at grid **interfaces**

```
porInt    <- porLiquid$int
diffDIC   <- diffHCO3 / (1-log(porInt^2))
```

tortuosity correction

Step 4: Implementation in R (ReacTran)

4.3 Define model parameters that are **constant** throughout the spatial domain

```
parms <- c(
 Dbio      = 5/365,      # bioturbation mixing coefficient
 v_adv     = 0.005,      # sediment advection velocity
 rMin      = 0.01,       # POC mineralisation rate constant
 depoPOC   = 100,        # POC deposition rate at SWI
 bwDIC     = 2000        # DIC concentration at SWI
)
```

Universiteit Utrecht

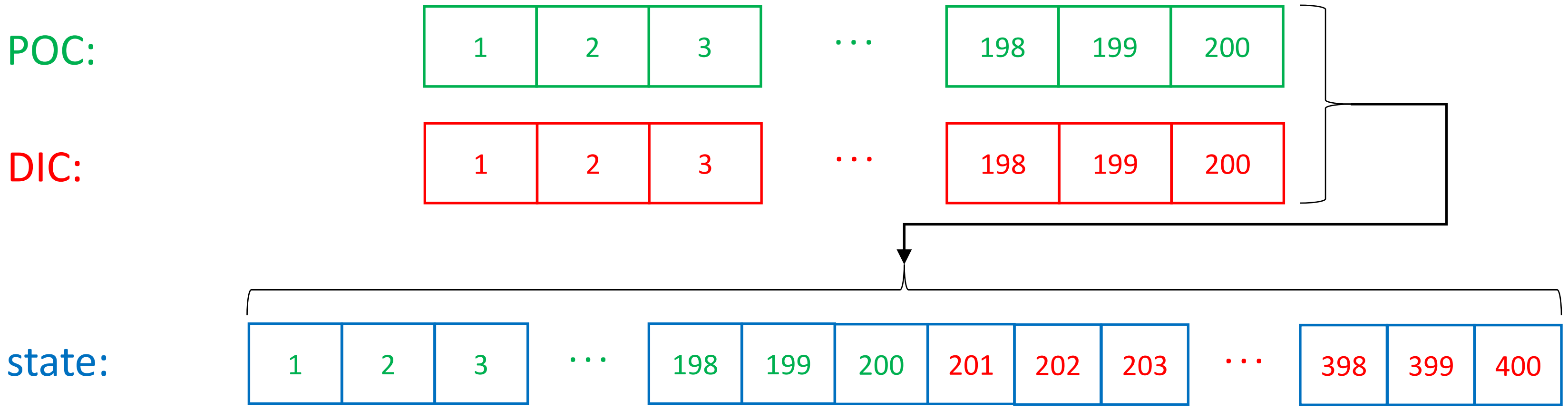
GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.4 Define and initialize the vector of **state variables**

```
names     <- c("POC", "DIC")
nspec     <- length(names)
POC.ini   <- rep(0, length = N)
DIC.ini   <- rep(0, length = N)
```

```
state     <- c(POC.ini, DIC.ini)
```

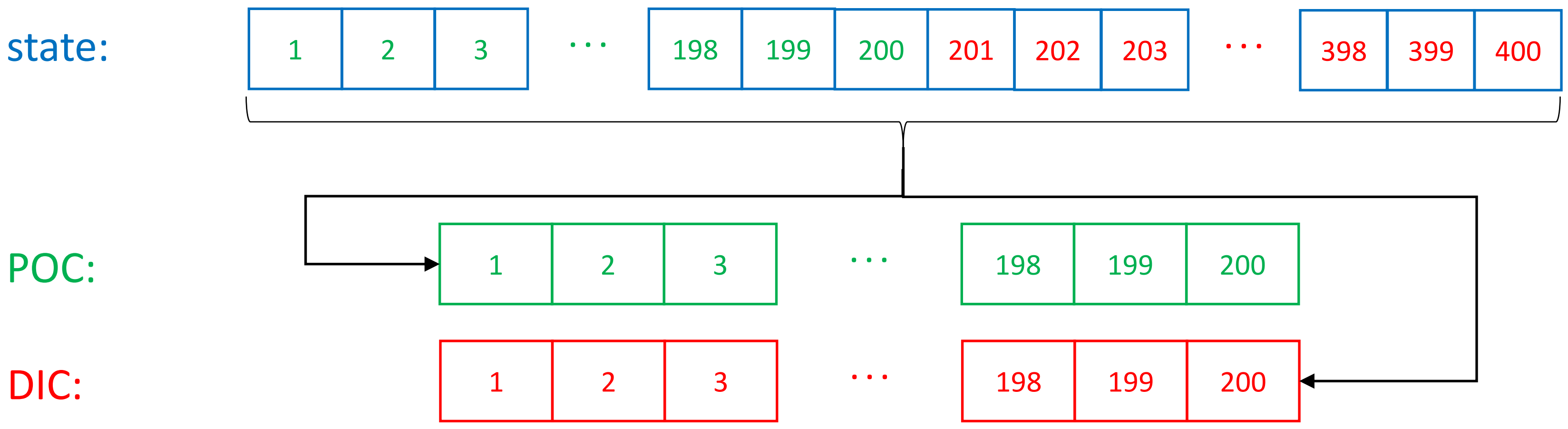


Step 4: Implementation in R (ReacTran)

4.5 Define model function

```
Diamodel <- function (t, state, pars)
{
  with (as.list(pars),{
       POC <- state[   1  :   N ]
       DIC <- state[(N+1):(2*N)]
```

“**Extract**” values of state variables from the input!



Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.6 Calculate the **transport term** for each component

Condition at **upper** boundary

```
tran.POC <- tran.1D(C = POC, flux.up = depoPOC, dx = Grid,
                    VF = porSolid, D = Dbio, v = v_adv)
```

Volume fraction!
(**1-porosity** for POC)

Diffusion
coefficient

Advective
velocity

Condition at **upper** boundary

```
tran.DIC <- tran.1D(C = DIC, C.up = bwDIC, dx = Grid,
                    VF = porLiquid, D = diffDIC, v = v_adv)
```

Volume fraction!
(**porosity** for POC)

Diffusion
coefficient
(at grid interfaces!)

Advective
velocity

If not specified, boundary condition is **dC/dx = 0 by default**!

Universiteit Utrecht

GHENT
UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.7 Calculate **process rates**

```
Mineralisation <- rMin * POC
```

$$\underbrace{mol\ C\ m_s^{-3}\ d^{-1}}_{\texttt{Mineralisation}} \qquad \underbrace{d^{-1}}_{\texttt{rMin}} \qquad \underbrace{mol\ C\ m_s^{-3}}_{\texttt{POC}}$$

. . .

More rate expressions if more processes!

Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.8 Calculate **time-derivative** for each state variable: **dC/dt = transport + reaction**

```
dPOC.dt <- tran.POC$dC - Mineralisation
```

$mol\ C\ m_s^{-3}\ d^{-1}$ — Transport term — Net reaction term

Porosity values in the **middle** of grid boxes

```
poro    <- porLiquid$mid
dDIC.dt <- tran.DIC$dC + Mineralisation*(1-poro)/poro
```

$mol\ C\ m_L^{-3}\ d^{-1}$ — Transport term — Net reaction term converted to correct units!!

Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.9 Output **time-derivatives** combined into one long vector

```
return(list(c(dPOC.dt, dDIC.dt))
```

Order must be consistent with the “extraction” step in the **beginning** of the model function!!

```
POC <- state[    1  :    N ]
DIC <- state[(N+1):(2*N)]
```

Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.10 Output **additional** variables

Depth-profile of the process rate

```r
return(list(c(dPOC.dt, dDIC.dt) ,
        Mineralisation = Mineralisation ,
        TotalMin       = sum(Mineralisation*Grid$dx*porSolid$mid) ,
        DIC.SWI.Flux   = tran.DIC$flux.up ,
        DIC.Deep.Flux  = tran.DIC$flux.down ,
        POC.SWI.Flux   = tran.POC$flux.up ,
        POC.Deep.Flux  = tran.POC$flux.down
))
```

Depth-integrated mineralization rate $(mol\ C\ m^{-2}\ d^{-1})$

Fluxes at the domain boundaries $(mol\ C\ m^{-2}\ d^{-1})$

Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

4.11 Find **steady-state** solution

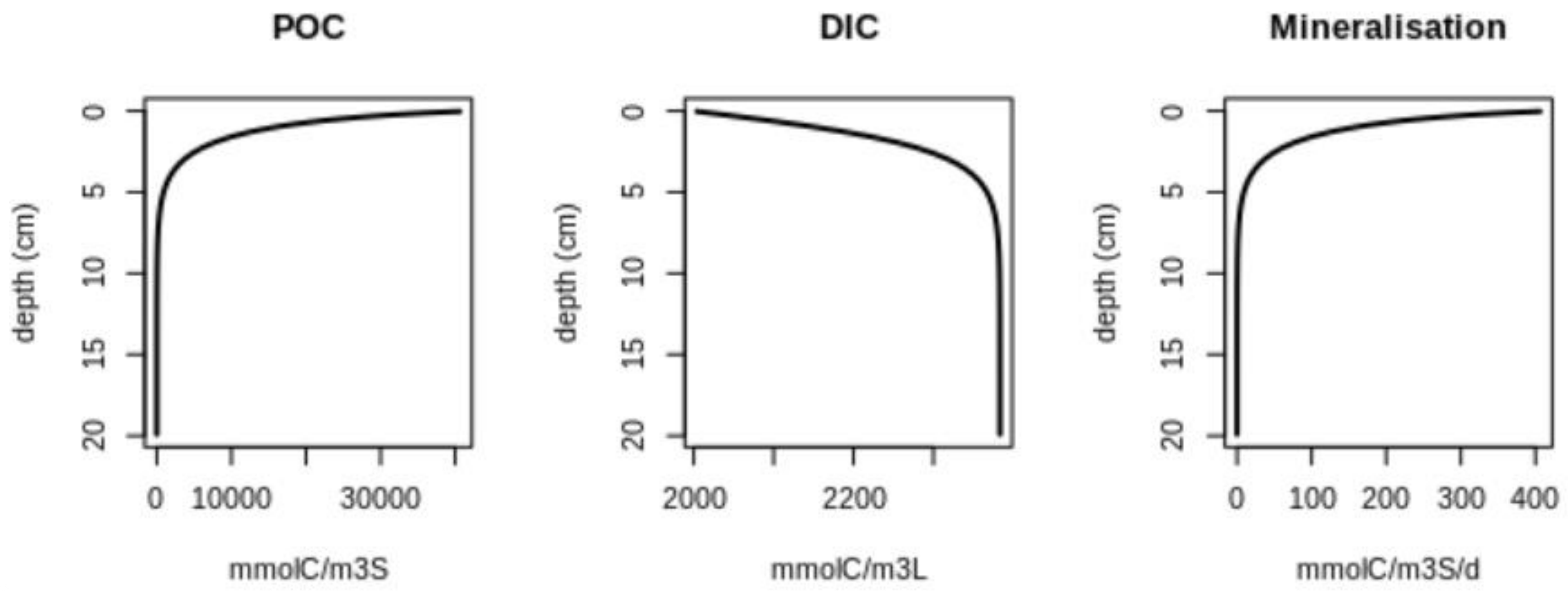
```
std  <- steady.1D(y = state, func = Diamodel, parms = parms,
                  nspec = nspec, dimens = N, names = names,
                  positive = TRUE)
```

Only find solution with **positive** values
(concentrations *cannot* be negative)

Step 4: Implementation in R (ReacTran)

4.12 Plot **steady-state** solution

```
plot(std, xyswap=TRUE, grid = Grid$x.mid, lty=1, lwd=2,
     which = c("POC", "DIC", "Mineralisation"),
     xlab=c("mmolC/m3S", "mmolC/m3L", "mmolC/m3S/d"),
     ylab="depth (cm)", mfrow=c(1,3))
```



Universiteit Utrecht

GHENT UNIVERSITY

Step 4: Implementation in R (ReacTran)

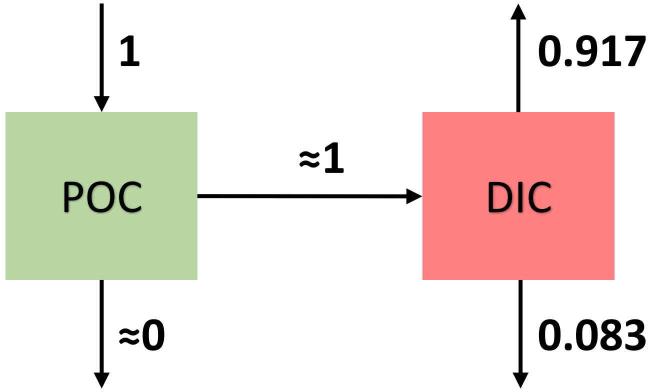
4.13 Construct **steady-state** carbon budget

```
toselect <- c("TotalMin", "POC.SWI.Flux", "POC.Deep.Flux",
        "DIC.SWI.Flux", "DIC.Deep.Flux")
BUDGET <- std[toselect]
unlist(BUDGET)
```

```
  TotalMin POC.SWI.Flux POC.Deep.Flux DIC.SWI.Flux DIC.Deep.Flux
  9.99e+01 1.000000e+02 7.816205e-05  -9.16583e+01 8.341621e+00
```

Conceptual box model:

All values in **mmol C m^{-2} d^{-1}**

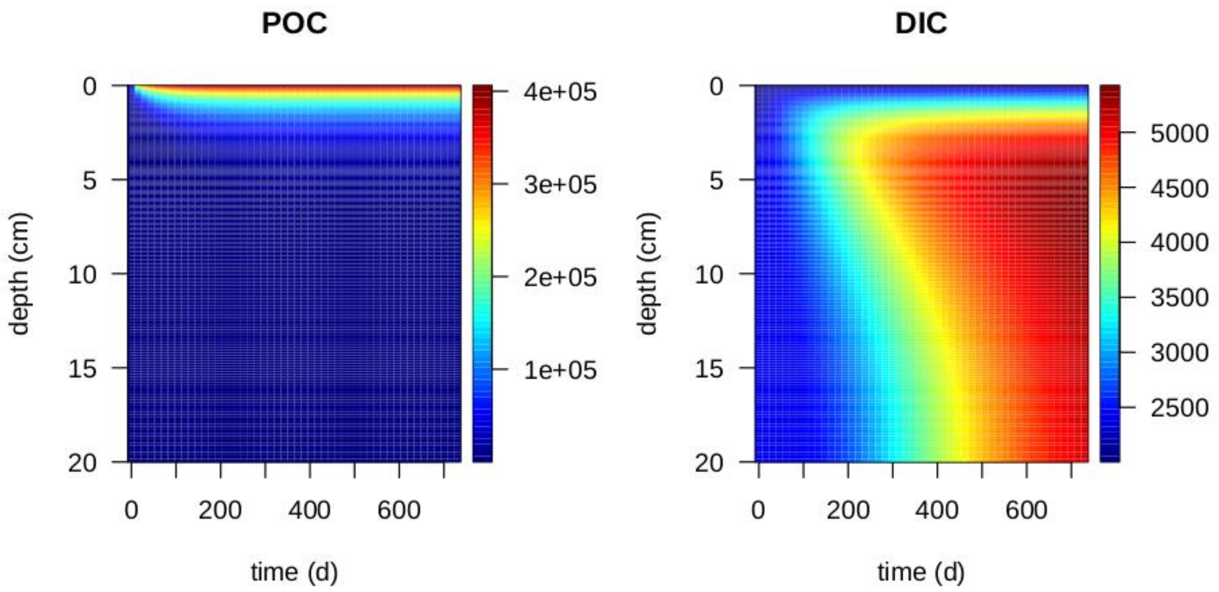


Step 4: Implementation in R (ReacTran)

4.14 Calculate and display a **dynamic** solution

```
times <- seq(from=0, to=2*365, length.out=50)
parms["depoPOC"] <- 1000
out <- ode.1D(y=std$y, func=Diamodel, parms=parms, times=times,
              nspec=nspec, dimens=N, names=names)
```

```
image(out, legend=TRUE, grid=Grid$x.mid, ylim=(Length,0),
           las=1, ylab="depth (cm)", xlab="time (d)")
```



Universiteit Utrecht

GHENT UNIVERSITY

Conclusion

Generic “skeleton” of a 1D ReacTran model:

1. Define spatial **grid** and model **parameters**
2. Define and initialize **state variables**
3. Define **model function** (transport and reaction terms, time-derivatives)
4. Calculate and display model **results**
5. Check **mass balances** and make **budget**

Most important functions: `setup.grid.1D` `setup.prop.1D`
`tran.1D` `steady.1D` `ode.1D`

Best approach : start with a **template** (RT1D_porous.Rmd)

and **expand** it based on your needs!

Universiteit Utrecht

GHENT UNIVERSITY

Reactive Transport in the Hydrosphere

Department of Earth Sciences, Faculty of Geosciences, Utrecht University

Lecturers: Lubos Polerecky and Karline Soetaert

Illustrations, narration and video editing: Renee Hageman
Additional contributions: Dries Bonte, University Ghent
Audio effects: mixkit.co

Universiteit Utrecht

GHENT UNIVERSITY