Oxygen Diagenesis

Karline Soetaert and Arthur Capet

July 2017

Set-up

We start with a simple oxygen diagenetic model. We need

1. to load librairies R-package ReacTran.

```
require(ReacTran, quietly = TRUE) # transport models in aquatic systems
require(marelac, quietly = TRUE)  # toolbox for aquatic sciences
```

```
##
## Attaching package: 'marelac'
```

```
## The following objects are masked from 'package:oce':
##
##     coriolis, gravity
```

2. to set a spatial framework

Here, a vertical 1D grid with a 100 levels over 10cm.

```
grid <- setup.grid.1D(N = 100, L = 10)
```

3. Model equations

We consider **diffusion** and constant **respiration** above a certain depth.

$$\frac{\partial O_2(z)}{\partial t} = \frac{\partial O_2(z)}{\partial t}|_{trasnport} - \gamma(z)$$

O_2 is a concentration in mmol m^{-3}_{liquid} and we have to consider porosity ϕ.

$$\frac{\partial O_2(z)}{\partial t} = \frac{1}{\phi}\frac{\partial}{\partial z}[\phi D \frac{\partial C}{\partial z}] - \gamma(z)$$

Boundary conditons are imposed as:

- constant concentration at the sediment-water interface

$$O_2(z=0) = O_{2,bottom}$$

- nul gradient at the lower cell

$$\frac{\partial O_2}{\partial z}|_{(z=sediment\ bottom)} = 0$$

```
O2model <- function (t, O2, p) {
  with (as.list(p), {
    # Transport term (using ReacTran routines)
    O2tran <- tran.1D(C = O2, C.up = O2BW, D = D, VF = porosity, dx = grid)
    # Respiration
    O2cons <- minrate*(grid$x.mid < mindepth)
    # the function returns the time derivative
    list(O2tran$dC - O2cons, O2cons = O2cons)
  })
}
```

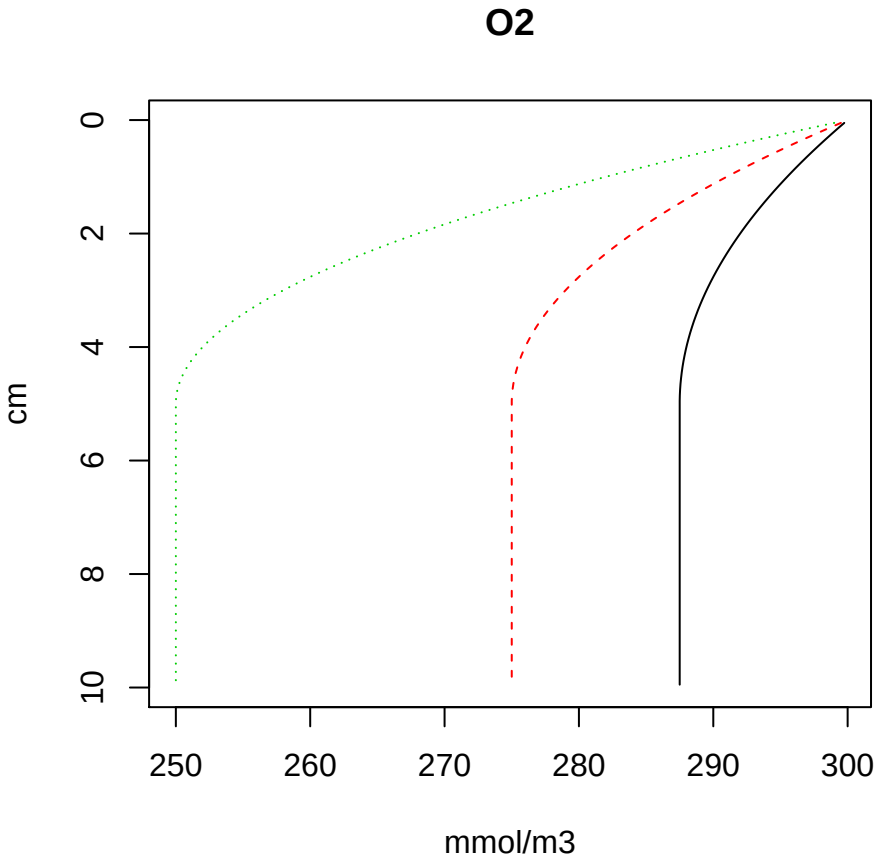
4. to assign a value to the model parameters

```
parms <- c(
  porosity = 0.8    , # -
  minrate  = 1      , # nmol O2/cm3/d - oxygen consumption rate
  mindepth = 5      , # cm            - depth below which minrate = 0
  O2BW     = 300    , # nmol/cm3       - bottom water oxygen concentration
  D        = 1        # cm2/d         - molecular diffusion coefficient
)
```

Simulation

Let us now compute the steady state solutions for three set of parameters (multiplying the respiration rate) and plot the ouputs.

```
parms2 <- parms3 <- parms
parms2["minrate"] <- 2
parms3["minrate"] <- 4

# random initial conditions
IC<-runif(length(grid$x.mid))

# computes the steay-state solution
out  <- steady.1D(y = IC, parms = parms, func = O2model, nspec = 1, names = "O2")
out2 <- steady.1D(y = IC, parms = parms2, func = O2model, nspec = 1, names = "O2")
out3 <- steady.1D(y = IC, parms = parms3, func = O2model, nspec = 1, names = "O2")

plot(out, out2, out3, xyswap = TRUE, xlab = "mmol/m3", ylab = "cm", grid = grid$x.mid)
```



Specific Outputs

So far the variable ´out´ contains the steady-state solution and the diagnostic ´O2cons´

```
str(out)
```

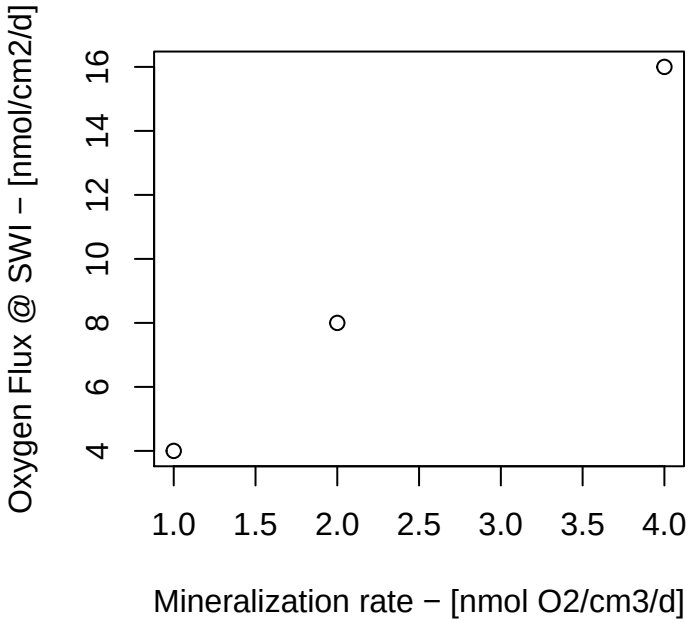
```
## List of 2
##  $ y     : num [1:100, 1] 300 299 299 298 298 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "O2"
##  $ O2cons: num [1:100] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "precis")= num [1:3] 6.52e+02 5.01e-03 2.13e-09
##  - attr(*, "steady")= logi TRUE
##  - attr(*, "class")= chr [1:3] "steady1D" "rootSolve" "list"
##  - attr(*, "dimens")= num 100
##  - attr(*, "nspec")= num 1
##  - attr(*, "ynames")= chr "O2"
```

The model declaration is modified to add a diagnostic : the oxygen flux at the sediment-water interface ´O2flux´.

```
O2model <- function (t, O2, p) {
  with (as.list(p), {
    O2tran <- tran.1D(C = O2, C.up = O2BW, D = D, VF = porosity, dx = grid)
```

```
    O2cons <- minrate*(grid$x.mid < mindepth)
    list(O2tran$dC - O2cons, O2cons = O2cons,O2flux=O2tran$flux.up)
  })
}
```

```
out  <- steady.1D(y = IC, parms = parms, func = O2model, nspec = 1, names = "O2")
out2 <- steady.1D(y = IC, parms = parms2, func = O2model, nspec = 1, names = "O2")
out3 <- steady.1D(y = IC, parms = parms3, func = O2model, nspec = 1, names = "O2")

plot(x = c(parms["minrate"] , parms2["minrate"] , parms3["minrate"]), y = c(out$O2flux , out2$O2flux, ou
     xlab="Mineralization rate - [nmol O2/cm3/d]" , ylab="Oxygen Flux @ SWI - [nmol/cm2/d]")
```



At steady-state, the flux at the interface equals the total amount of oxygen consumed in the sediment. Here, consumption takes place only in the liquid, so we have to consider porosity in the integral.

$$\int_{\infty}^{0} \phi\gamma \; dz = \phi\gamma_0 L$$

SensRange

The library FME is a toolbox for exploring model characteristics.

```
library(FME)
```

```
## Loading required package: coda
```

We will only use the function sensRange, that allows to visualize the variability of model outputs when a parameter is allowed to vary within a given range.

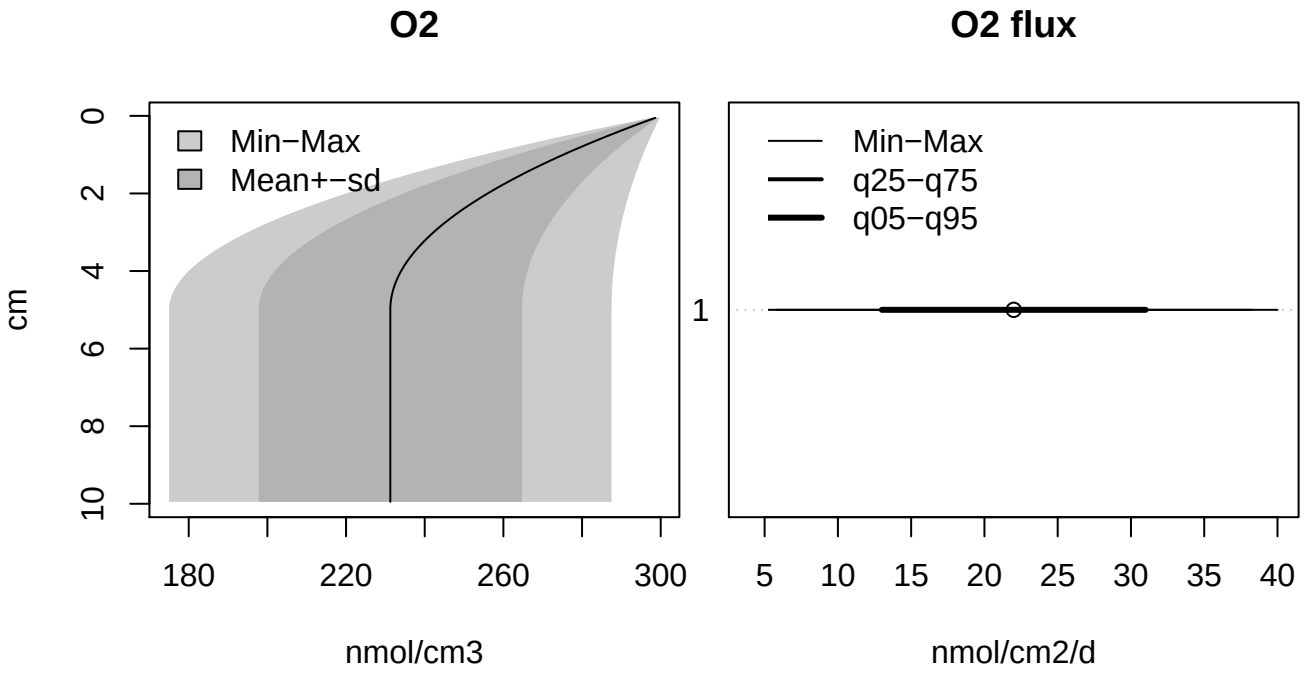
´sensRange´ needs

- a data frame to precise which parameter should vary and within which range.

```
parRange <- data.frame(min=1,max=10)
rownames(parRange)<-"minrate"
```

- A function that reecievs a set of parameter and returns the response variable to be investigated (here we define two functions, for ´O2´ and ´O2flux´)

```
solveO2<-function(pars){
  out  <- steady.1D(y = IC, parms = pars, func = O2model, nspec = 1, names = "O2")
  return(data.frame(Depth=grid$x.mid , O2=out$y))
}

solveO2f<-function(pars){
  out  <- steady.1D(y = IC, parms = pars, func = O2model, nspec = 1, names = "O2")
  return(data.frame(Depth=0,O2flux=out$O2flux))
}


sR<-sensRange(func = solveO2, sensvar = "O2", parms=parms,
              parRange = parRange, num=50, dist='grid')

sRf<-sensRange(func = solveO2f, sensvar = "O2flux", parms=parms,
               parRange = parRange, num=50, dist='grid')

plot(summary(sR),xyswap = TRUE, xlab = "nmol/cm3", ylab = "cm")
plot(summary(sRf), xlab = "nmol/cm2/d",  main="O2 flux")
```



Exercices

1. Explore the range of model response to variations of other parameters

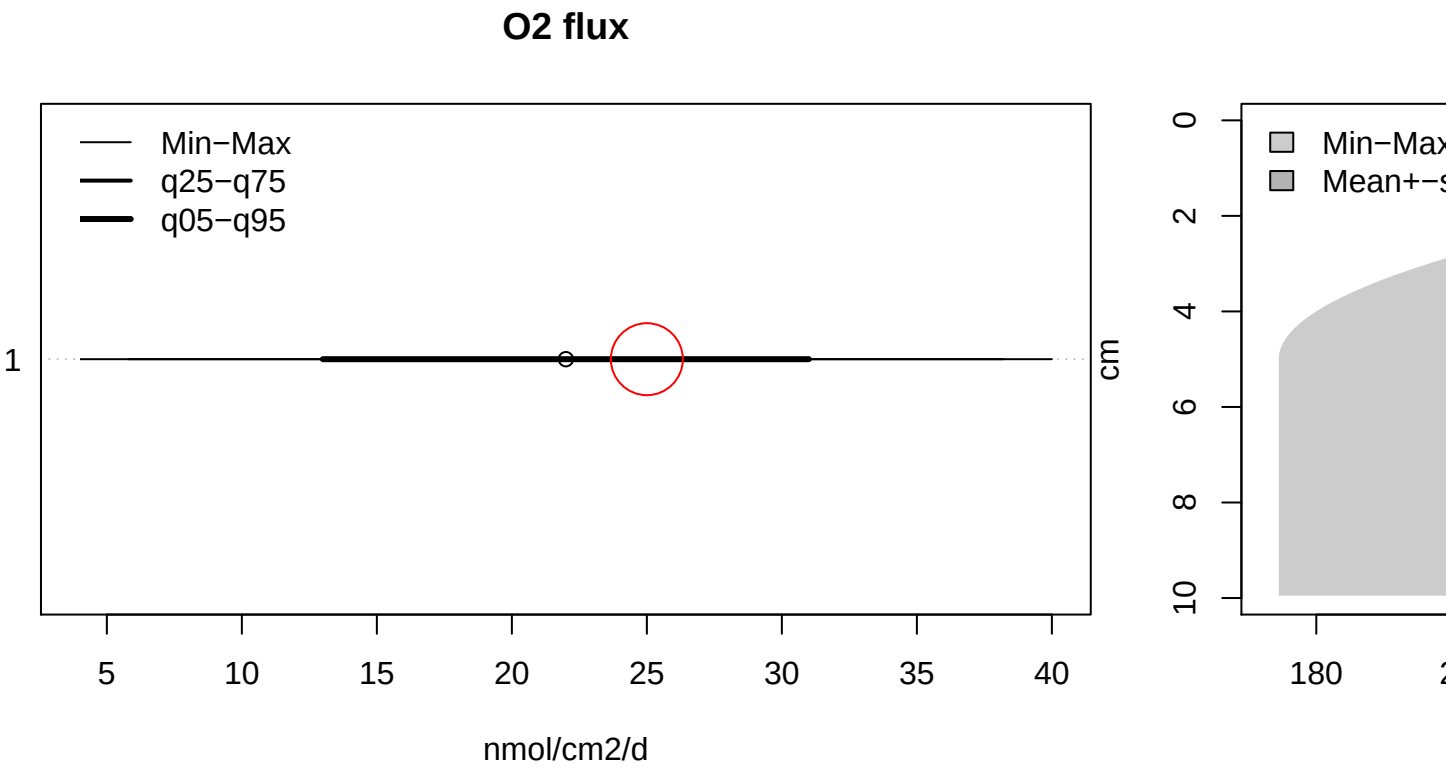
Hint : Just adapt parRange in the upper code.

2. Read measured data (Flux and/or Profiles), include them to the plot

Hint : use read.table to read the data. Add them to the previous plot using ´line´ and ´points´

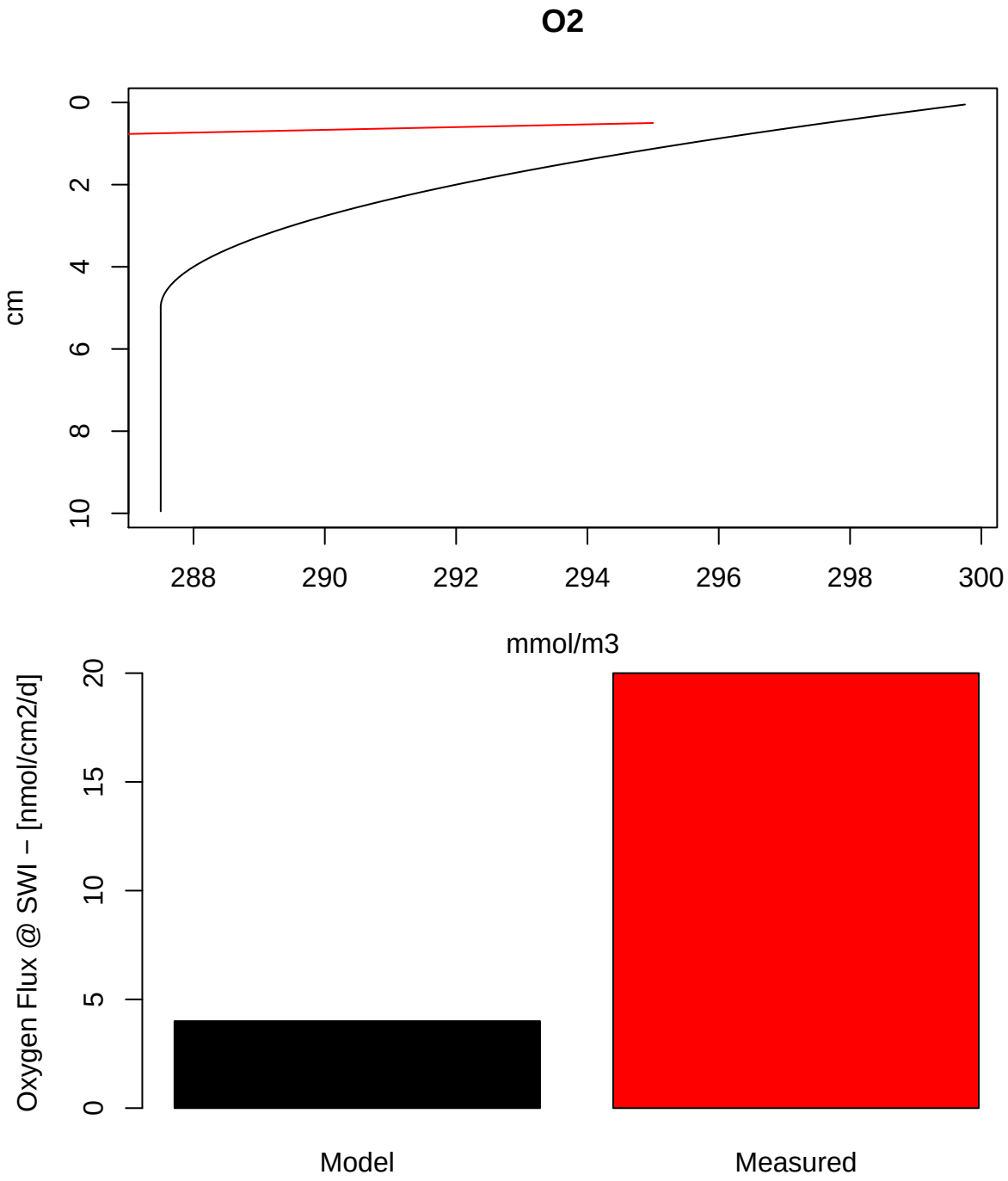
Example:

```
measuredO2flux = 20

O2prof<-matrix(ncol=2,  data=c(0.5,1,2,3,4,285,280,270,260,255))
colnames(O2prof)<-c("depth","O2")

plot(summary(sRf), xlab = "nmol/cm2/d", main="O2 flux")
points(x= 25,y=1, col="red", cex=5 )

plot(summary(sR),xyswap = TRUE, xlab = "nmol/cm3", ylab = "cm")
lines(y=O2prof[,'depth'] ,x=O2prof[,'O2'], col="red" )
```



Can you try parameters value that match your measurements ?

```r
plot(out, xyswap = TRUE, xlab = "mmol/m3", ylab = "cm", grid = grid$x.mid)
lines(y=c(0.5,1,2,3,4) ,x=c(295,280,270,260,255), col="red" )

barplot( height = c(out$O2flux, measuredO2flux ), ylab="Oxygen Flux @ SWI - [nmol/cm2/d]", names.arg = c
```



Same with the other parameters, also testing different distributions

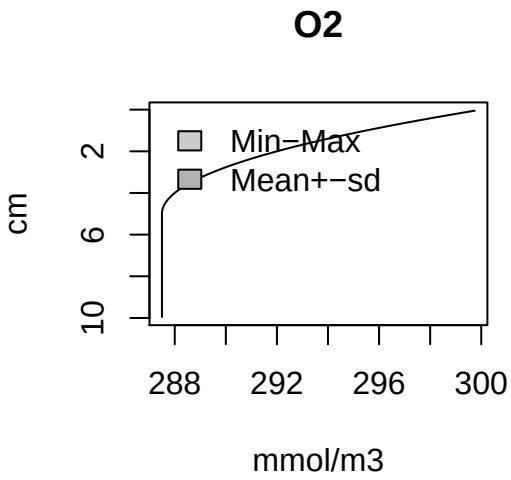
“‘r library(FME)

parRange <- data.frame(min= c(.5 , 1 , 1 , 200 , .1), max= c(1 , 10 , 10 , 400 , 10)) rownames(parRange)<- names(parms) “‘

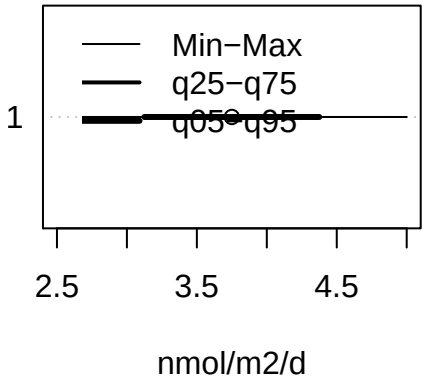
Table 1: The parameters bounds for sensitivity.

	min	max
porosity	0.5	1
minrate	1.0	10
mindepth	1.0	10
O2BW	200.0	400
D	0.1	10

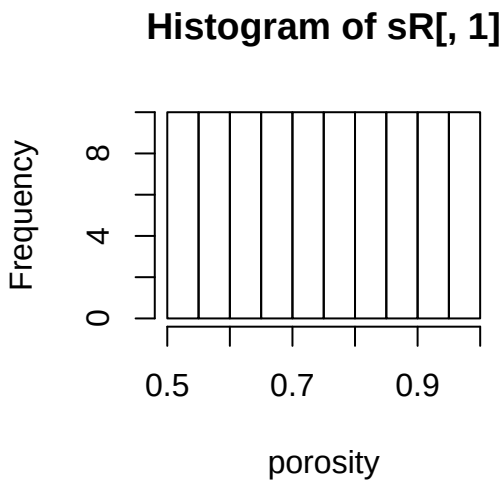
```
par(mfrow=c(2,2))

parRangelocal<-parRange[1,]

sR<-sensRange(func = solveO2, sensvar = "O2", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
sRf <- sensRange(func = solveO2f, sensvar = "O2flux", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
plot(summary(sR),xyswap = TRUE, xlab = "mmol/m3", ylab = "cm")
```



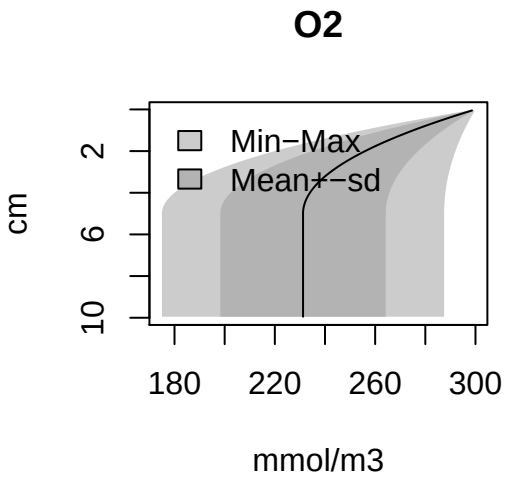

```
plot(summary(sRf),xyswap = TRUE, xlab = "nmol/m2/d", ylab = "cm")
```



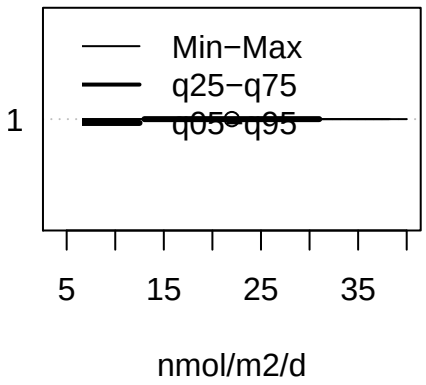
```
hist(sR[,1], xlab = rownames(parRangelocal))
```



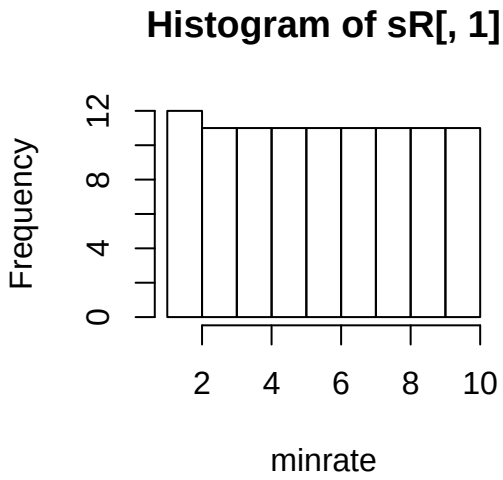
```
parRangelocal<-parRange[2,]

sR<-sensRange(func = solveO2, sensvar = "O2", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
sRf <- sensRange(func = solveO2f, sensvar = "O2flux", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
plot(summary(sR),xyswap = TRUE, xlab = "mmol/m3", ylab = "cm")
```



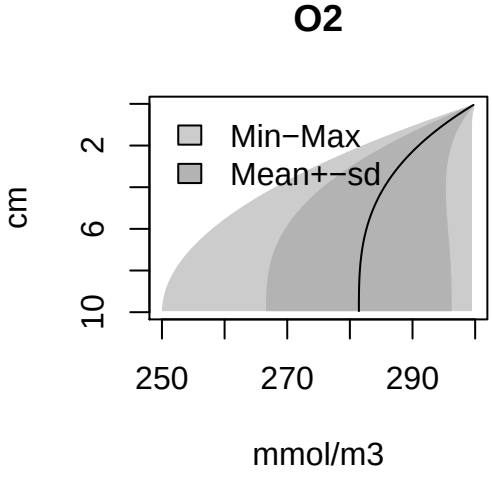

```
plot(summary(sRf),xyswap = TRUE, xlab = "nmol/m2/d", ylab = "cm")
```



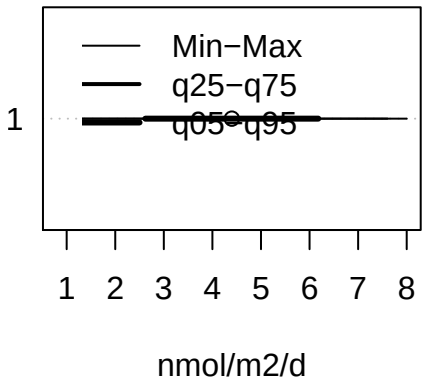

```
hist(sR[,1], xlab = rownames(parRangelocal))
```



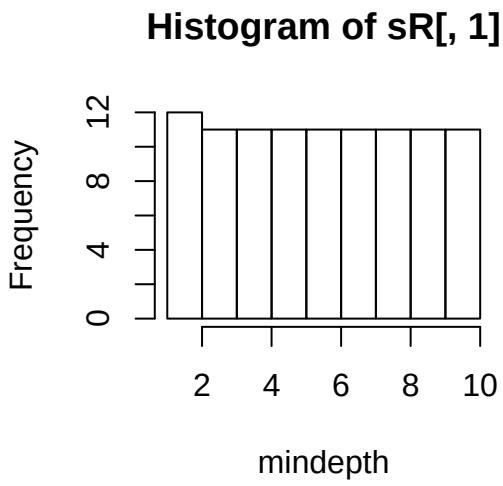
```
parRangelocal<-parRange[3,]
sR<-sensRange(func = solveO2, sensvar = "O2", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
sRf <- sensRange(func = solveO2f, sensvar = "O2flux", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
plot(summary(sR),xyswap = TRUE, xlab = "mmol/m3", ylab = "cm")
```



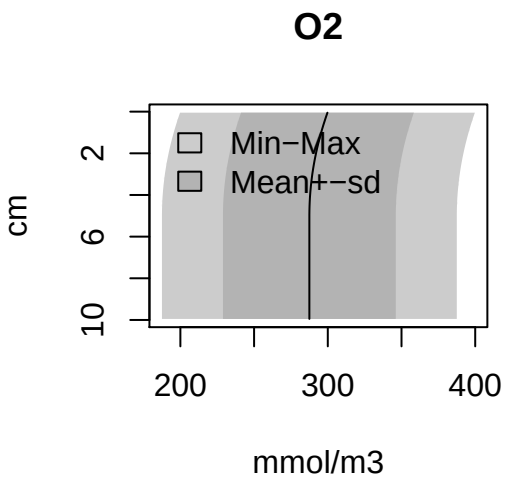
```
plot(summary(sRf),xyswap = TRUE, xlab = "nmol/m2/d", ylab = "cm")
```



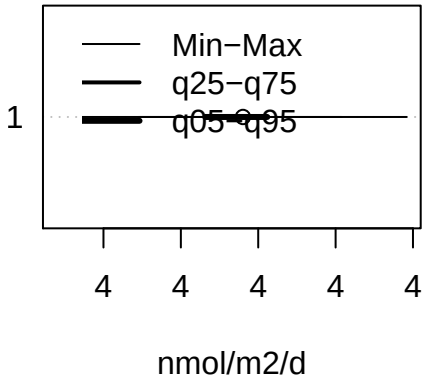

```
hist(sR[,1], xlab = rownames(parRangelocal))
```



```
parRangelocal<-parRange[4,]

sR<-sensRange(func = solveO2, sensvar = "O2", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
sRf <- sensRange(func = solveO2f, sensvar = "O2flux", parms=parms, parRange = parRangelocal,
              num=100, dist='grid')
plot(summary(sR),xyswap = TRUE, xlab = "mmol/m3", ylab = "cm")
```




```
plot(summary(sRf),xyswap = TRUE, xlab = "nmol/m2/d", ylab = "cm")
```




```
hist(sR[,1], xlab = rownames(parRangelocal))
```