femmeR – An R companion to FEMME

Henrik Andersson   Andreas Hofmann   Karline Soetaert

February 27, 2006

femmeR is a package of functions mainly used to visualize output from the ecological modelling environment FEMME developed by Soetaert et al. [2002].

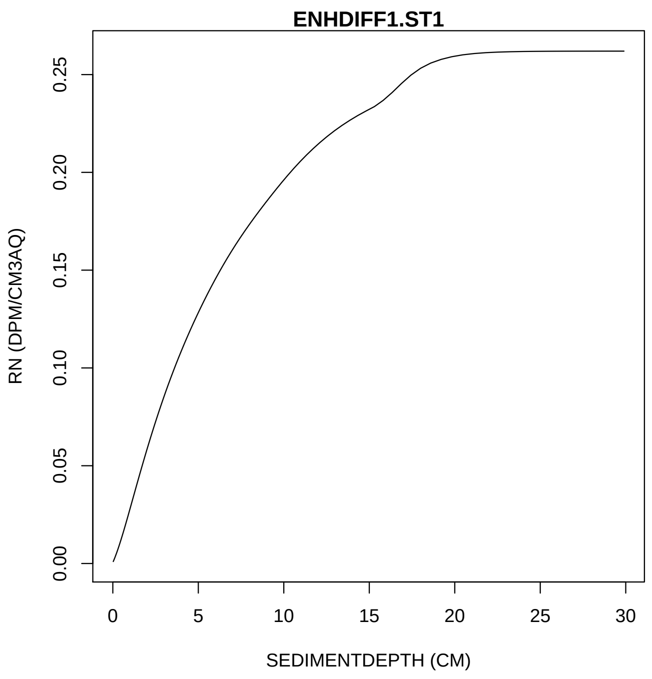
Contents

This document explains the use of femmeR, however if you are not familiar with R a brief guide is found in Appendix A

1. Steady state 1D

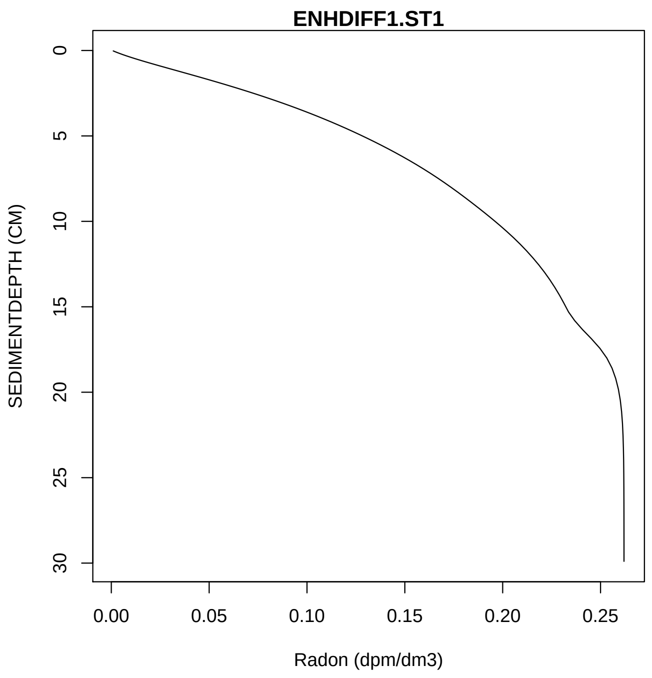
To process the output from a steady-state application, read the file and plot the results,

```
> test <- read.st1("ENHDIFF1.ST1")
> plot(test)
```



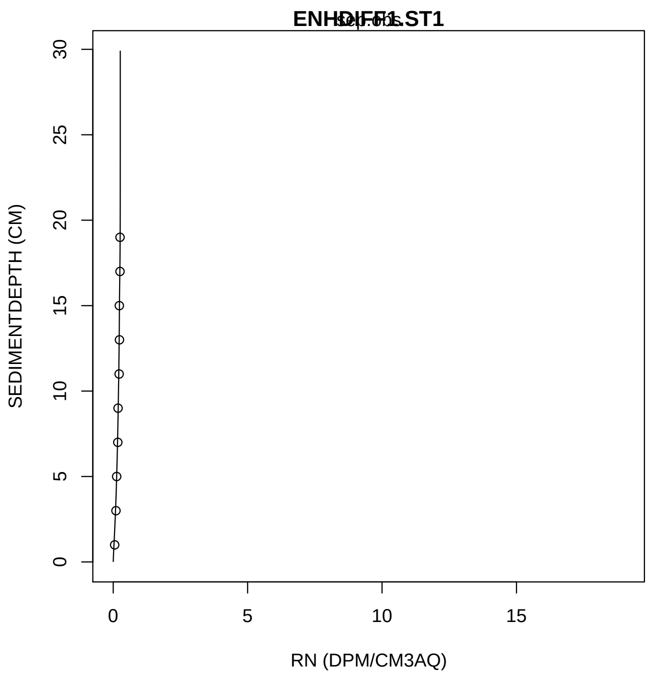
The default plot command chooses the first and second variable in the output file, to plot different variables, change `xvari` or `yvari` and reverse an axis using e.g. `rev="y"`, and change the default labels.

```
> plot(test, xvari = 2, yvari = 1, rev = "y", xlab = "Radon (dpm/dm3)")
```



Adding observed values to the output is very useful to check the model performance,

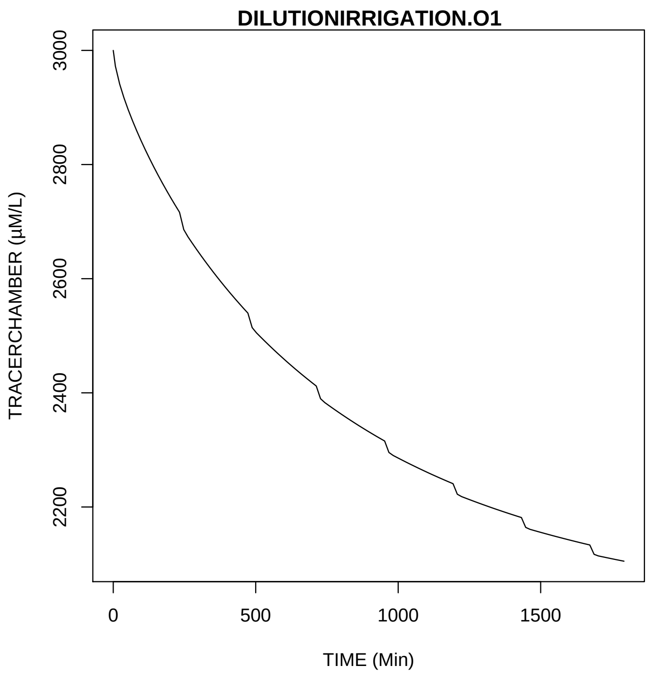
```
> test.obs <- read.obs("sep.obs")
> plot(test, xvari = 1, yvari = 2, rev = "vx", obs = test.obs)
```



2. Time dependent 0-d

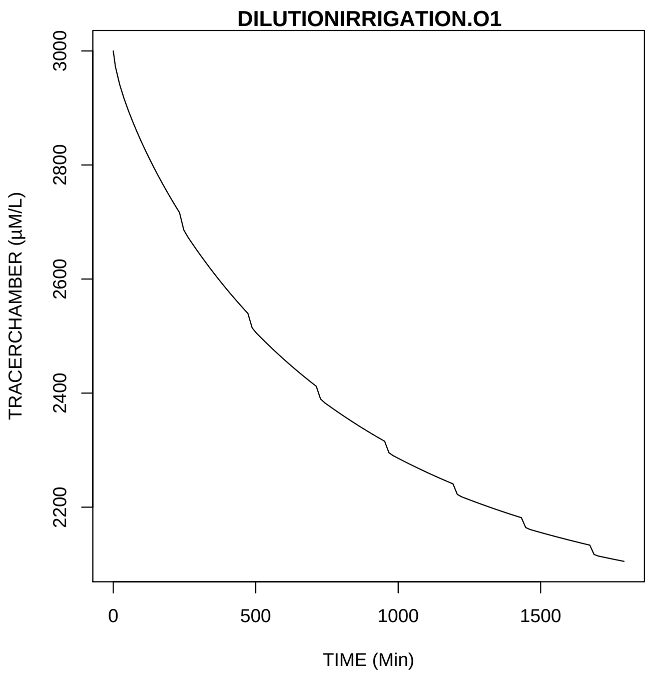
Time dependent 0-d output are read and plotted in almost exactly the same way as one-dimensional steady-state output.

```
> test.o1 <- read.o1("DILUTIONIRRIGATION.O1")
> plot(test.o1, xvar = 1, yvar = 3)
```



Variable can be selected either using the number in which they appear in the file or by a string with the variable name,

```
> plot(test.o1, xvari = "TIME", yvari = 3)
```

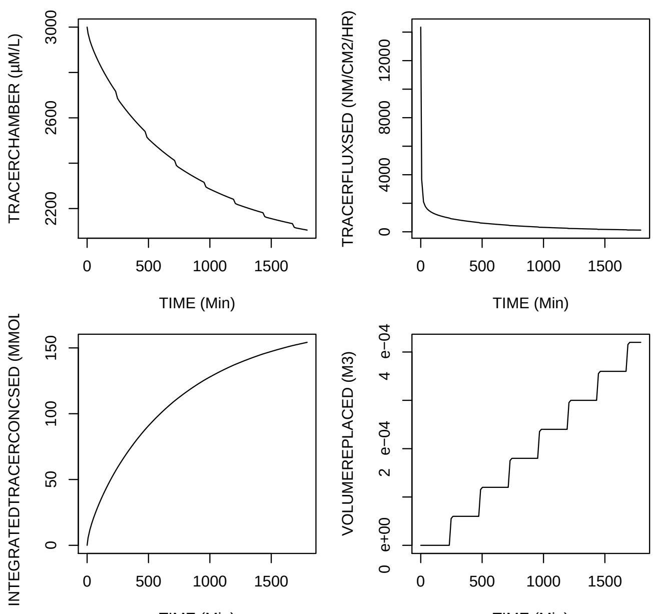


If you don't know which variables you have, or simply forgot the clever names you gave them:

```
> test.o1

femmeR object of class o1
Filename: DILUTIONIRRIGATION.O1

Variables
--------------------
1 TIME
2 SURFACEMIXINGRATE
3 TRACERCHAMBER
4 TRACERCHAMBERINI
5 TRACERFLUXSED
6 TRACERDEEPFLUX
7 INTEGRATEDTRACERCONCCHA
8 INTEGRATEDTRACERCONCSED
9 INTEGRATEDTRACERCONC
10 VOLUMEREPLACED
11 TRACERREMOVED
12 CHAMBERVOLUME
13 INTEGRATEDCONSUMPTION
```

Plotting several variables at once is also possible, selecting a set with e.g. `yvari=c(3,5,8)` or `yvari=2:5`, or using the actual names of the variables, `yvari=c("PH","CO2")`.

```
> plot(test.o1, xvar = 1, yvari = c(3, 5, 8, 10), main = "")
```



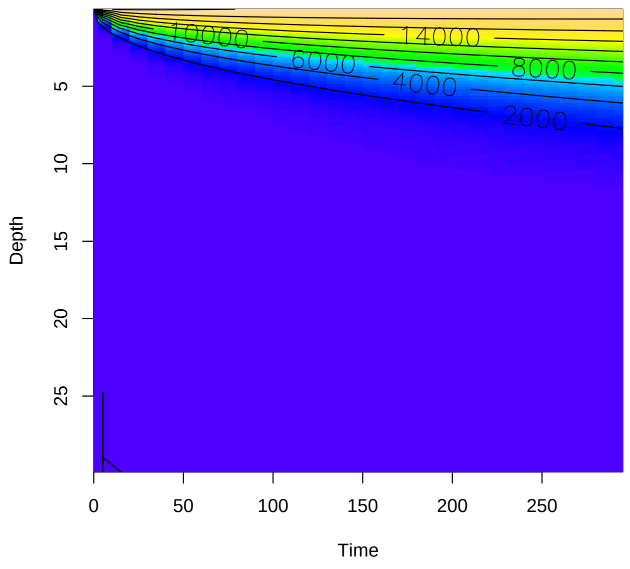
3. Time dependant 1-d

Reading a file with data varying in space and time is done in the same way as for other files.

```
> test.o2 <- read.o2("BERG.02")
```

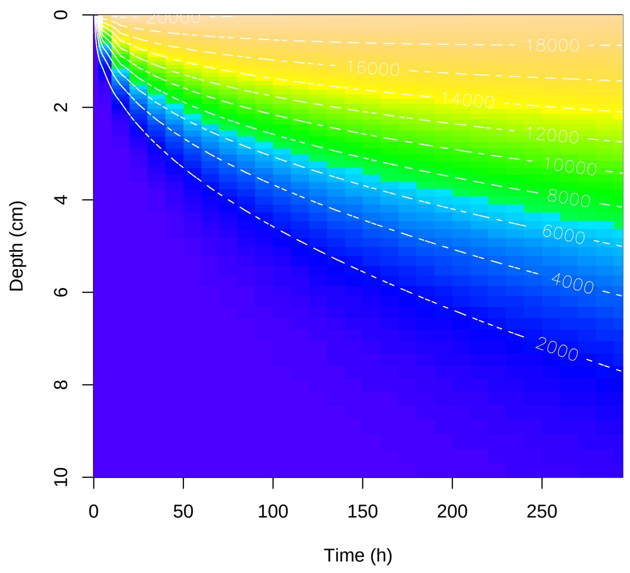
Visualizing xyz data can be done in numerous ways, the simplest is to use the function `plot.o2`:

```
> plot(test.o2, zvari = 3, rev = "y")
```



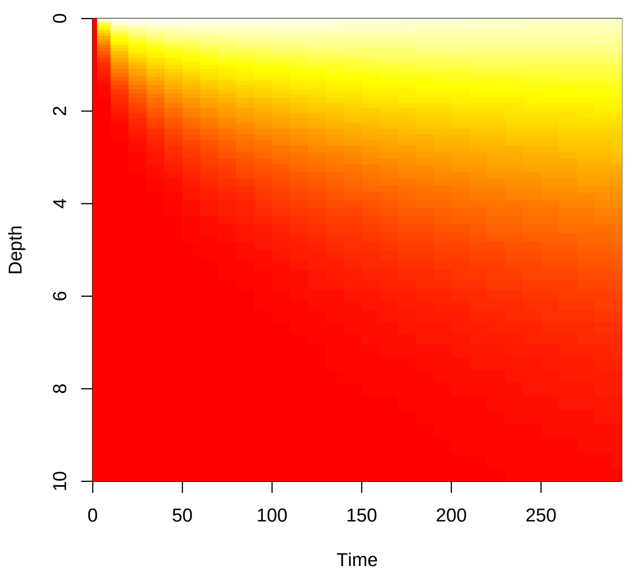
As usual a number of options can be set, axis labels and contour lines can be changed:

```
> plot(test.o2, zvari = 3, ylim = c(10, 0), ylab = "Depth (cm)",
+     xlab = "Time (h)", linecol = "white", labcex = 1, lty = 2)
```



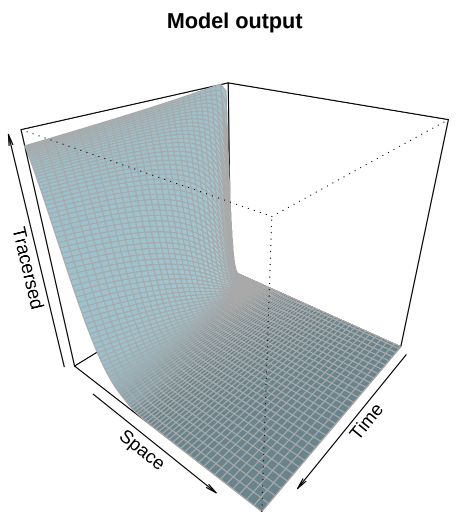
Or if you don't like the contourlines and like other color palettes better:

```
> plot(test.o2, zvari = 3, ylim = c(10, 0), col = heat.colors(50),
+     contour = FALSE)
```



You can also use R functions directly:

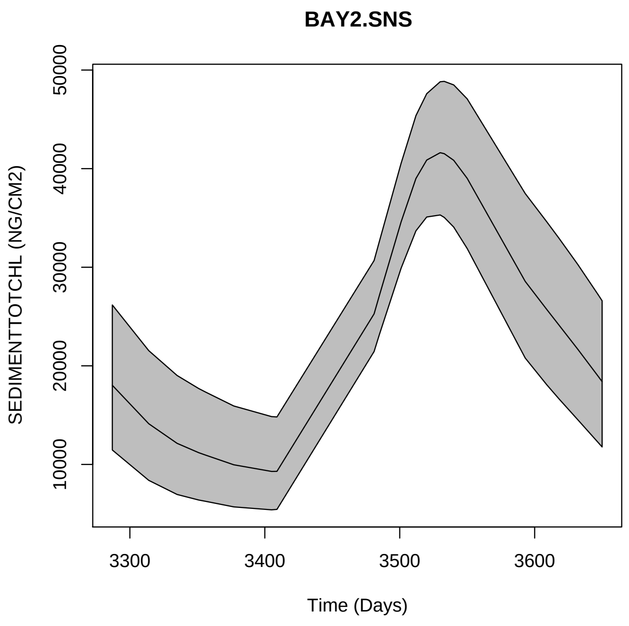
```
> x <- test.o2$time
> y <- test.o2$depth
> z <- test.o2$data$TRACERSED
> persp(x, y, z, theta = 130, phi = 30, xlab = "Time", ylab = "Space",
+     zlab = "Tracersed", shade = 0.7, main = "Model output", ,
+     col = "lightblue", border = "darkgray")
```



4. Sensitivity analysis

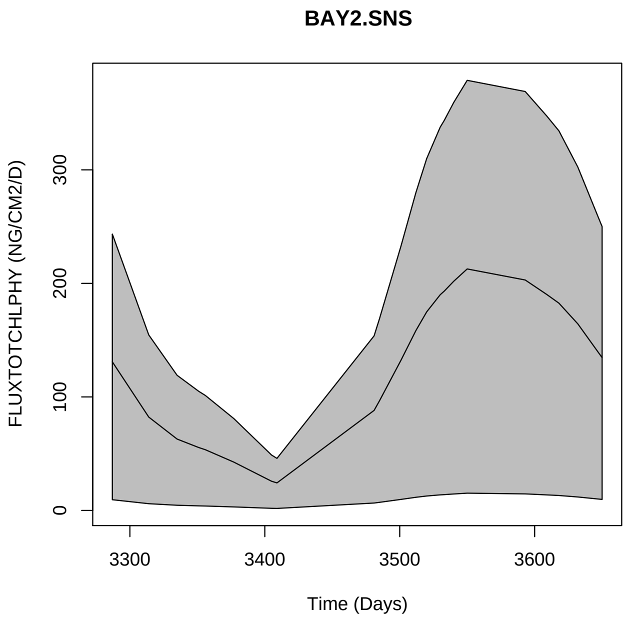
Read the file and plot it

```
> sensible <- read.sns("BAY2.SNS")
> plot(sensible)
```



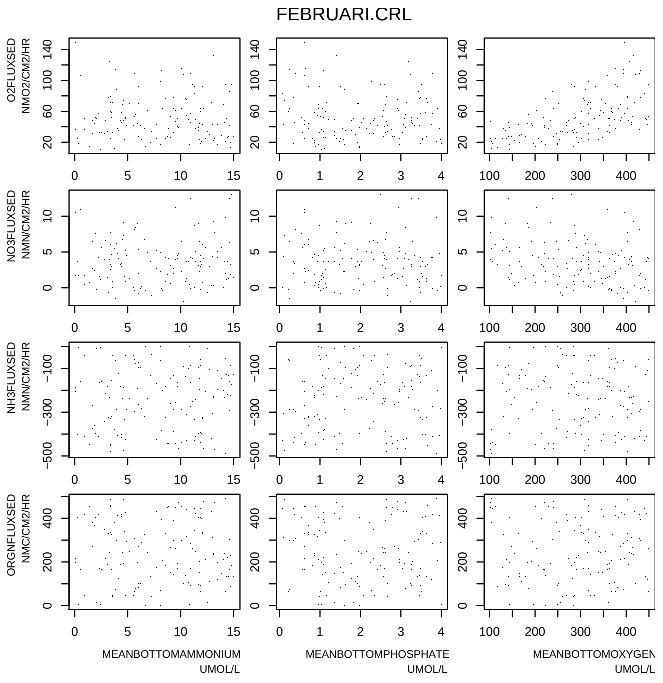
Or another variable . . .

```
> plot(sensible, yvari = 6)
```



5. Monte Carlo simulations

```
> feb.crl <- read.crl("FEBRUARI.CRL")
> plot(feb.crl, xvari = 3:5, yvari = 25:28, size = 0.6)
```



6. Parameter Covariance output analysis

Read the file created by FEMME with

```
> deepradon <- read.pcv("DEEPCOLLIN.PCV")
```

Sensitivity values can be summarized using the following indexes,

$$\delta_j^{msqr} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} s_{i,j}^2} = \frac{1}{\sqrt{n}}||\mathbf{s}_j||, \tag{1}$$

$$\delta_j^{mabs} = \frac{1}{n}\sum_{i=1}^{n} |s_{i,j}|, \tag{2}$$

$$\delta_j^{mean} = \frac{1}{n}\sum_{i=1}^{n} s_{i,j}, \tag{3}$$

$$\delta_j^{max} = \max_i s_{i,j}, \tag{4}$$

$$\delta_j^{max} = \min_i s_{i,j}. \tag{5}$$

using,

```
> summary(deepradon)

                        dmsqr        dmabs          mean           max
SURFACEPOROSITY  0.0132193601 0.0076389698 -0.0076389698 -5.587935e-09
MIXINGLAYER      0.0084282839 0.0036822949 -0.0036822949 -1.044571e-06
IRRIGATIONFACTOR 0.0082134266 0.0053855233 -0.0053337942  3.879685e-04
DEEPPOROSITY     0.0066735393 0.0046760328 -0.0030379979  7.975223e-03
IRRIGATIONRATE   0.0063249306 0.0050384581 -0.0050384581 -1.458824e-05
POROSITYCOEFF    0.0014904050 0.0008433500 -0.0008182440  1.882954e-04
DBCOEFF          0.0008639463 0.0004119066 -0.0004119066 -6.584451e-08
                          min
SURFACEPOROSITY  -0.031942436
MIXINGLAYER      -0.029093511
IRRIGATIONFACTOR -0.016922771
DEEPPOROSITY     -0.014398609
IRRIGATIONRATE   -0.010037018
POROSITYCOEFF    -0.003582213
DBCOEFF          -0.002560170
```

This is already done by FEMME, but using R we can easily select only a few columns or combine results from several runs,

```
> summary(deepradon)[, 1:2]

                        dmsqr        dmabs
SURFACEPOROSITY  0.0132193601 0.0076389698
MIXINGLAYER      0.0084282839 0.0036822949
IRRIGATIONFACTOR 0.0082134266 0.0053855233
DEEPPOROSITY     0.0066735393 0.0046760328
IRRIGATIONRATE   0.0063249306 0.0050384581
POROSITYCOEFF    0.0014904050 0.0008433500
DBCOEFF          0.0008639463 0.0004119066
```

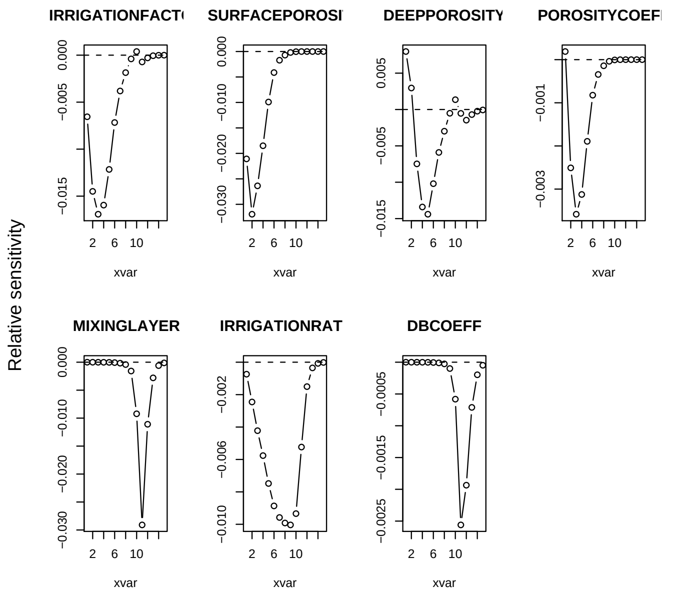
To make this into a LaTeX table the `xtable` package is useful,

```
> library(xtable)

> xtable(summary(deepradon), digits = rep(4, 6))
```

To make a plot with default settings, type,

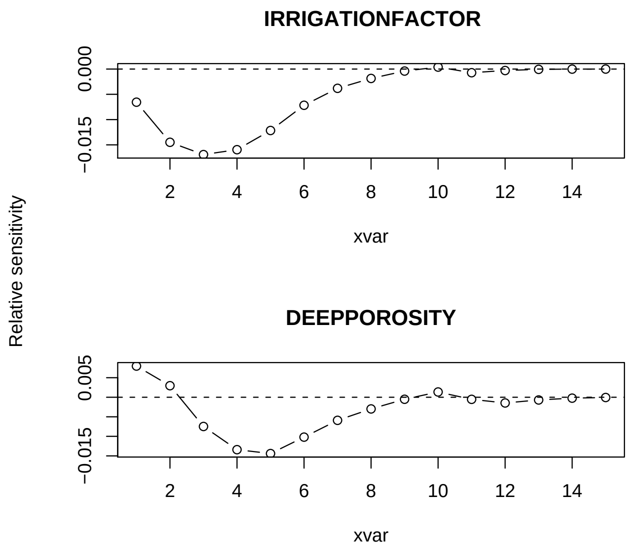
```
> plot(deepradon)
```

	dmsqr	dmabs	mean	max	min
SURFACEPOROSITY	0.0132	0.0076	−0.0076	−0.0000	−0.0319
MIXINGLAYER	0.0084	0.0037	−0.0037	−0.0000	−0.0291
IRRIGATIONFACTOR	0.0082	0.0054	−0.0053	0.0004	−0.0169
DEEPPOROSITY	0.0067	0.0047	−0.0030	0.0080	−0.0144
IRRIGATIONRATE	0.0063	0.0050	−0.0050	−0.0000	−0.0100
POROSITYCOEFF	0.0015	0.0008	−0.0008	0.0002	−0.0036
DBCOEFF	0.0009	0.0004	−0.0004	−0.0000	−0.0026



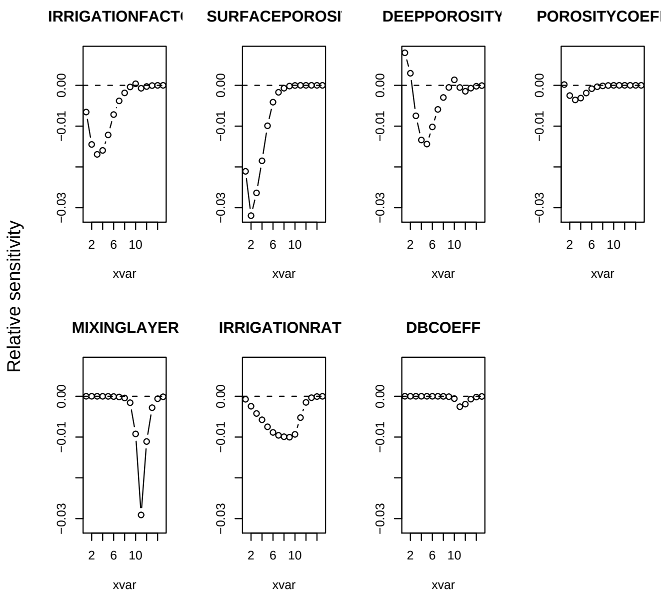
It is also possible to plot only one or a selection of the parameters,

```
> plot(deepradon, pari = c(1, 3))
```



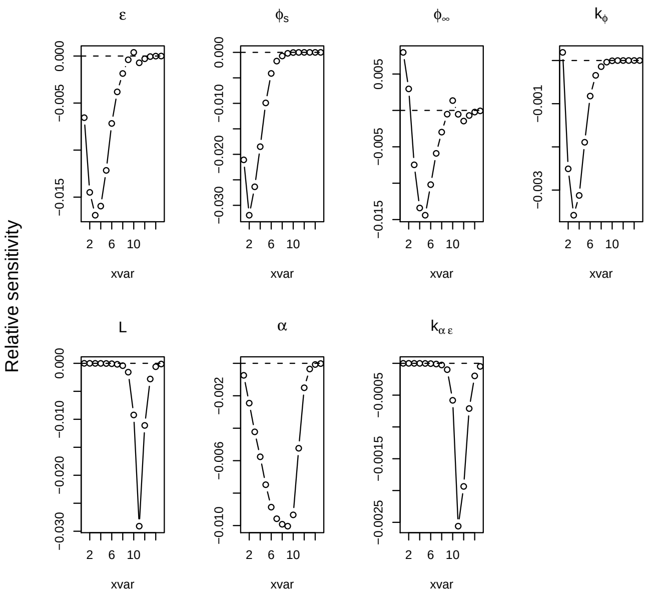
Or to use a common Y-axis to get the relative importance of the parameters in a graphical way,

```
> plot(deepradon, scale = TRUE)
```



For publishing quality figures you might want the parameter names to be different than the names defined in the Fortran code,

```
> deepradon.parnames <- c(expression(epsilon), expression(phi[s]),
+     expression(phi[infinity]), expression(k[phi]), expression(L),
+     expression(alpha), expression(k[alpha ~ epsilon]), )
> plot(deepradon, parnames = deepradon.parnames)
```



```
> deepradon.parnames.tex <- c("$\\epsilon$", "$\\phi_s$", "$\\phi_\\infty$",
+     "$k_\\phi$", "$L$", "$\\alpha$", "$k_{\\alpha,\\epsilon}$",
+     )
> xtable(summary(deepradon, parnames = deepradon.parnames.tex),
+     digits = rep(4, 6))
```

	dmsqr	dmabs	mean	max	min
ϕ_s	0.0132	0.0076	−0.0076	−0.0000	−0.0319
L	0.0084	0.0037	−0.0037	−0.0000	−0.0291
ϵ	0.0082	0.0054	−0.0053	0.0004	−0.0169
ϕ_∞	0.0067	0.0047	−0.0030	0.0080	−0.0144
α	0.0063	0.0050	−0.0050	−0.0000	−0.0100
k_ϕ	0.0015	0.0008	−0.0008	0.0002	−0.0036
$k_{\alpha,\epsilon}$	0.0009	0.0004	−0.0004	−0.0000	−0.0026

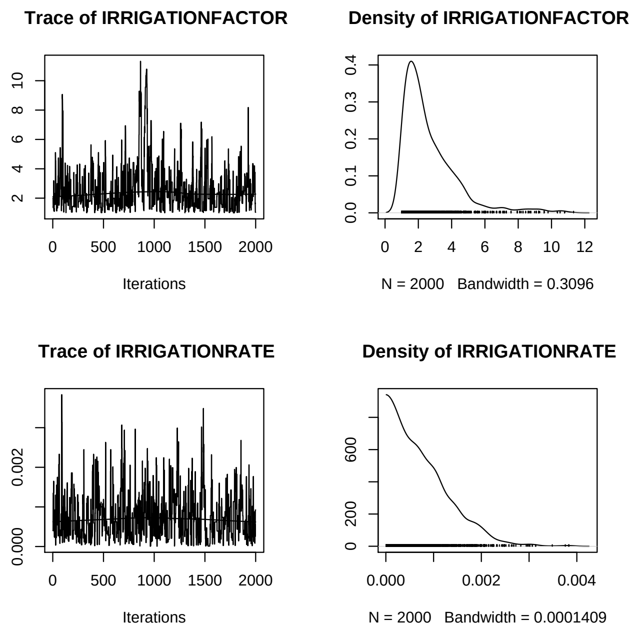
7. Bayesian data analysis

Handling bayesian output is done in R by package coda [Plummer et al., 2005]. femmeR only provides the ability to read a file from Application = BAYES. This creates a list with three components, filename, vars and data, where data is the MCMC values.

```
> berg <- read.bay("BERG_END.BAY")
> summary(berg$data)

Iterations = 1:2000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                      Mean       SD  Naive SE Time-series SE
IRRIGATIONFACTOR 2.6622577 1.663231 3.719e-02      1.494e-01
IRRIGATIONRATE   0.0007625 0.000608 1.359e-05      3.655e-05

2. Quantiles for each variable:

                      2.5%       25%      50%      75%    97.5%
IRRIGATIONFACTOR 1.0268510 1.5159075 2.123330 3.305668 7.942050
IRRIGATIONRATE   0.0000248 0.0002663 0.000626 0.001088 0.002156
```

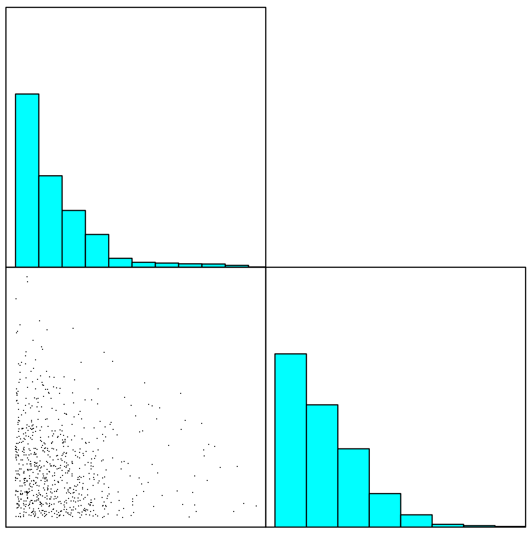
To have a look at how the MCMC was progressing and also to get an idea of what the posterior distribution looks like it can be visualized in two different ways, first like this:

```
> plot(berg)
```



And secondly as a pairs plot:

```
> pairs(berg)
```



To check if the MCMC simulation was run long enough we can run one of the diagnostic utilities from `coda`.

```
> raftery.diag(berg$data, r = 0.0125)
```

```
Quantile (q) = 0.025
Accuracy (r) = +/- 0.0125
Probability (s) = 0.95

                  Burn-in  Total Lower bound  Dependence
                  (M)      (N)   (Nmin)       factor (I)
 IRRIGATIONFACTOR 25       4297  600          7.16
 IRRIGATIONRATE   23       3921  600          6.54
```

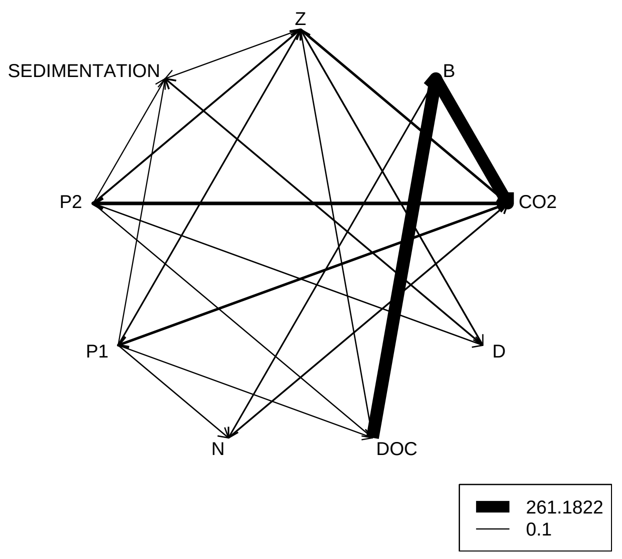
Apparently our 2000 runs are not long enough, so according to Raftery and Lewis [1996] we tune the jumping distribution by calculating the conditional standard deviation of the parameters multiply these with 2.3 and set this as the new jumping distribution.

```
> par1 <- lm(berg$data[, 1] ~ berg$data[, -1])
> summary(par1)$sigma

[1] 1.650772

> par2 <- lm(berg$data[, 2] ~ berg$data[, -2])
> summary(par2)$sigma

[1] 0.0006034033
```

8. Inverse analysis

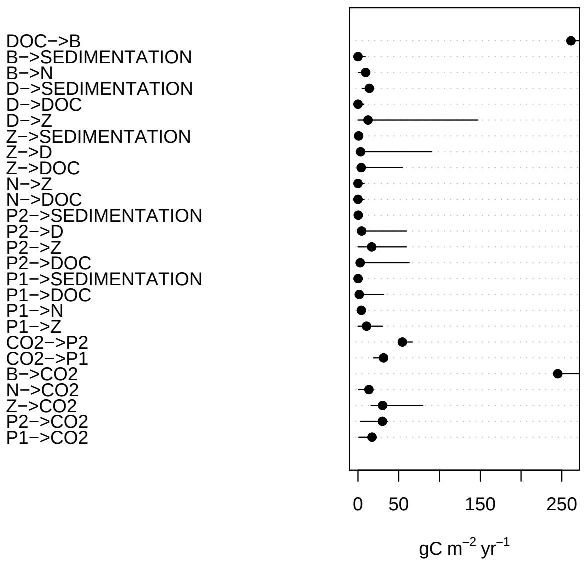
Food webs calculated by inverse analysis:

```
> donali <- read.web("DONALI.WEB")
> plot(donali, sizelab = 1)
```



Ranges of all flows:

```
> dotchart.web(donali, xlab = expression(gC ~ m^{
+     -2
+ } ~ yr^{
+     -1
+ }))
```



9. Parameter files

It can be useful to be able to read and write parameter files if parameters are varied systematically from within R and changes in model output are visualized. It can also be useful for logging purposes.

```
> august <- read.par("august.par")
> august

Parameter values for tracer model
Testing, testing
##############
IrrigationType  =  BOTH
IrrigationFactor  =  8.13
IrrigationRate  =  4.8E-05
HeightChamber  =  0.105
SurfaceChamber  =  0.007853982
TracerAdded  =  14.43
SedimentationRate  =  0.
TemperatureChamber  =  16
SurfacePorosity  =  0.516
DeepPorosity  =  0.443
PorosityCoeff  =  0.3
MixingLayer  =  10.
DbCoeff  =  1.

> august$DeepPorosity

[1] "0.443"

> august[[2]] <- 5
> write.par(august, file = "")

Parameter values for tracer model
Testing, testing
##############
IrrigationType  =  BOTH
IrrigationFactor  =  5
IrrigationRate  =  4.8E-05
HeightChamber  =  0.105
SurfaceChamber  =  0.007853982
TracerAdded  =  14.43
SedimentationRate  =  0.
TemperatureChamber  =  16
SurfacePorosity  =  0.516
```

```
DeepPorosity  =  0.443
PorosityCoeff  =  0.3
MixingLayer  =  10.
DbCoeff  =  1.
```

```
> write.par(august, file = "test.par", ask = FALSE)
```

Please note that using write.par with file set to something else than "" and ask=F, will overwrite existing files with the same name without asking before

A. A short introduction to R

A.1. Using packages e.g. femmeR

To load packages for R use the command `library`
and to load femmeR assuming you haven't done it already use `library(femmeR)`

A.2. Working directory

To see which directory you are currently working in, use `getwd`
and to change it use `setwd`
. Just remember to use forward slashes separating the directory names.

If you work with different projects in some folders on your computer you can also use the `.Rdata`
file which is saved at the end of the session if you answer yes on "Save workspace image". Doubleclick the `.Rdata` and R will start up in that folder and your previous workspace is restored.

A.3. Getting help

To get help about a function use `?function` e.g `?plot` gets you all the information about scatter plots. If you don't know the name of the function, try `help.search("useful phrase")`. To read the femmeR manual use `vignette("femmeR")`.

A.4. Entering data

To create a vector use the command `c`
.

```
> n = c(1, 5, 7)
> n
```

```
[1] 1 5 7
```

Some examples of sequences:

```
> x = 1:10
> x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> y = seq(0, 1, length = 11)
> y
```

```
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
> F = rep("A", 2)
> F
```

```
[1] "A" "A"
```

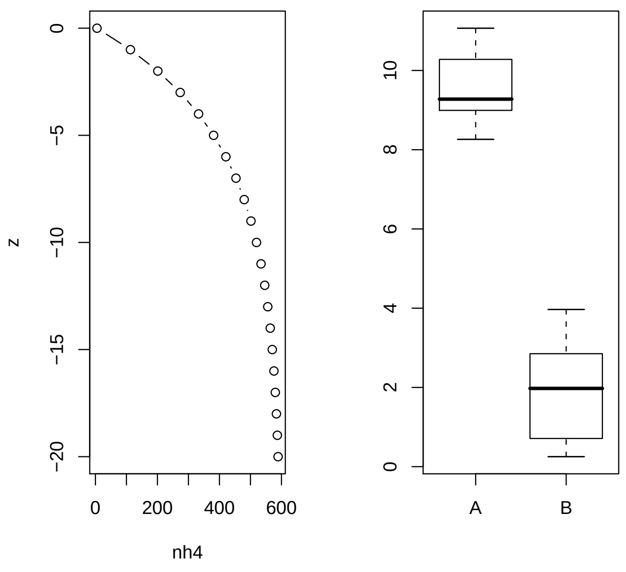
Combining them:

```
> G = rep(c("A", "B"), 3)
> G
```

```
[1] "A" "B" "A" "B" "A" "B"
```

A.5. Subplots

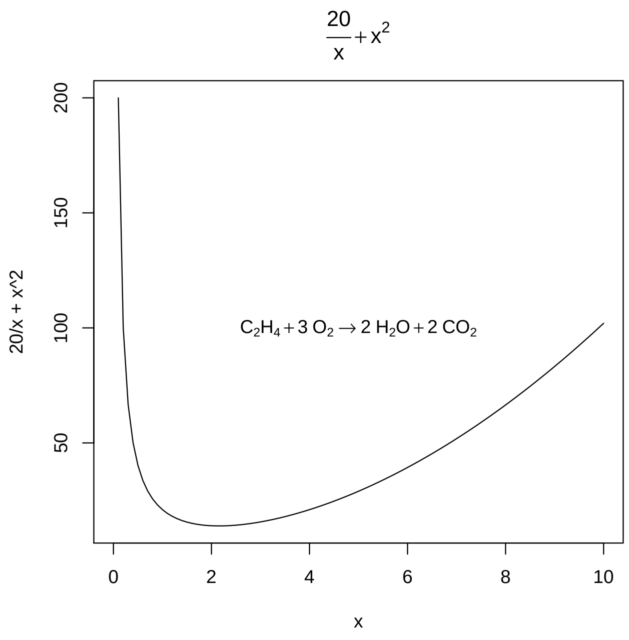
```
> par(mfrow = c(1, 2))
> plot(nh4, z, type = "b")
> boxplot(heffa ~ f)
```



A.6. Mathematical expressions

Here is a short example with some sub/super-scripts. To get more information, try `demo(plotmath)`.

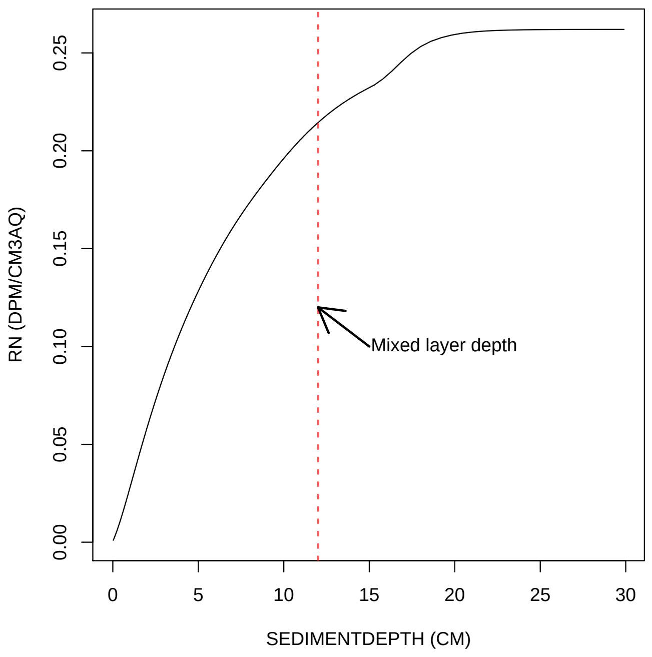
```
> x <- seq(0, 10, length = 100)
> y <- 20/x + x^2
> curve(20/x + x^2, xlim = c(0, 10))
> title(main = expression(frac(20, x) + x^2))
> text(5, 100, expression(C[2] * H[4] + 3 ~ O[2] %->% 2 ~ H[2] *
+     O + 2 ~ CO[2]))
```



A.7. Customizing plots

To find possible changes to make to a plot look at `?par`.

```
> plot(test, yvari = 2, main = "")
> abline(v = 12, lty = 2, col = 2)
> text(15.1, 0.1, "Mixed layer depth", adj = 0)
> arrows(15, 0.1, 12, 0.12, lwd = 2)
```

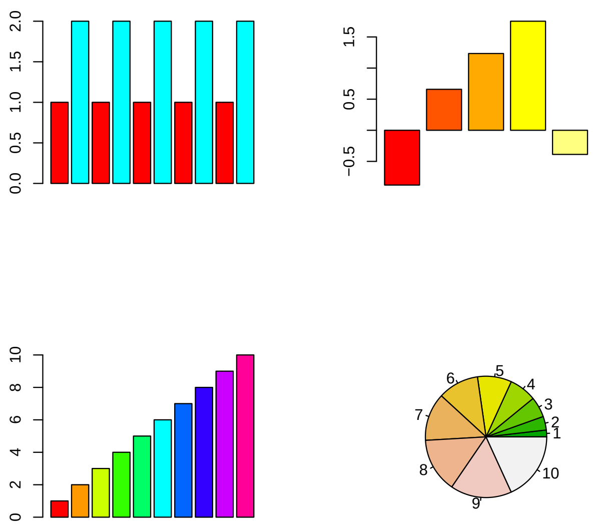


A.8. Using colors

If you make a graph intended for a poster or presentation, using colors can be very helpful and also makes your graphs look more interesting.

You can use colors by their name e.g. “green” or by hexadecimal notation. The latter is best generated by specialized functions such as `rainbow`.

```
> par(mfrow = c(2, 2))
> barplot(rep(c(1, 2), 5), col = rainbow(2))
> barplot(rnorm(5), col = heat.colors(5))
> barplot(1:10, col = rainbow(10))
> pie(1:10, col = terrain.colors(10))
```



References

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. *coda: Output analysis and diagnostics for MCMC*, 2005. URL `http://www-fis.iarc.fr/coda/`. R package version 0.9-2.

Adrian E Raftery and Steven M Lewis. Implementing mcmc. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, *Markov chain Monte Carlo in practice*, pages 115–130. Chapman, London, 1996.

K Soetaert, V deClippele, and P Herman. Femme, a flexible environment for mathematically modelling the environment. *Ecological Modelling*, 151:177–193, 2002. URL `http://www.nioo.knaw.nl/cemo/femme`.

Index