R-package marelac : utilities for the MArine, Riverine, Estuarine, LAcustrine and Coastal sciences

Karline Soetaert
NIOO-CEME
The Netherlands

Thomas Petzoldt
Technische Universität Dresden
Germany

Filip Meysman
VUB
Belgium

Abstract

Rpackage **marelac** (Soetaert, Petzoldt, and Meysman 2008) contains chemical and physical constants and functions, routines for unit conversion, and other utilities useful for MArine, Riverine, Estuarine, LAcustrine and Coastal sciences.

Keywords: marine, riverine, estuarine, lacustrine, coastal science, utilities, constants, R.

1. Introduction

R-package **marelac** has been designed as a tool for use by scientists working in the MArine, Riverine, Estuarine, LAcustrine and Coastal sciences.

It contains:

- chemical and physical constants, e.g. atomic weights, gas constants.
- conversion factors, e.g. gram to mol to liter conversions.
- functions, e.g. to estimate concentrations of conservative substances as a function of salinity, gas transfer coefficients, diffusion coefficients, ...

2. constants

2.1. AtomicWeight

```
> unlist(AtomicWeight)
```

```
        H         He         Li         Be          B          C          N
 1.007940   4.002602   6.941000   9.012182  10.811000  12.010700  14.006700
        O          F         Ne         Na         Mg         Al         Si
15.999400  18.998403  20.179700  22.989769  24.305000  26.981539  28.085500
        P          S         Cl         Ar          K         Ca         Sc
```

```
 30.973762  32.065000  35.453000  39.948000  39.098300  40.078000  44.955912
       Ti          V         Cr         Mn         Fe         Co         Ni
 47.867000  50.941500  51.996100  54.938045  55.845000  58.933195  58.693400
       Cu         Zn         Ga         Ge         As         Se         Br
 63.546000  65.409000  69.723000  72.640000  74.921600  78.960000  79.904000
       Kr         Rb         Sr          Y         Zr         Nb         Mo
 83.798000  85.467800  87.620000  88.905850  91.224000  92.906380  95.940000
       Tc         Ru         Rh         Pd         Ag         Cd         In
       NA 101.070000 102.905500 106.420000 107.868200 112.411000 114.818000
       Sn         Sb         Te          I         Xe         Cs         Ba
118.710000 121.760000 127.600000 126.904470 131.293000 132.905452 137.327000
       La         Ce         Pr         Nd         Pm         Sm         Eu
138.905470 140.116000 140.907650 144.242000         NA 150.360000 151.964000
       Gd         Tb         Dy         Ho         Er         Tm         Yb
157.250000 158.925350 162.500000 164.930320 167.259000 168.934210 173.040000
       Lu         Hf         Ta          W         Re         Os         Ir
174.967000 178.490000 180.947880 183.840000 186.207000 190.230000 192.217000
       Pt         Au         Hg         Tl         Pb         Bi         Po
195.084000 196.966569 200.590000 204.383300 207.200000 208.980400         NA
       At         Rn         Fr         Ra         Ac         Th         Pa
       NA         NA         NA         NA         NA 232.038060 231.035880
        U         Np         Pu         Am         Cm         Bk         Cf
238.028910         NA         NA         NA         NA         NA         NA
       Es         Fm         Md         No         Lr         Rf         Db
       NA         NA         NA         NA         NA         NA         NA
       Sg         Bh         Hs         Mt         Ds         Rg
       NA         NA         NA         NA         NA         NA
```

```
> AtomicWeight$H
```

```
[1] 1.00794
```

```
> (W_H2O<- with (AtomicWeight, 2*H + O))
```

```
[1] 18.01528
```

2.2. Constants

```
> data.frame(cbind(acronym=names(Constants),
+               matrix(ncol=3,byrow=TRUE,data=unlist(Constants),
+               dimnames=list(NULL,c("value","units","description")))))
```

```
  acronym         value         units                 description
1       g           9.8          m/s2        gravity acceleration
2      SB    5.6697e-08     W/m^2/K^4   Stefan-Boltzmann constant
```

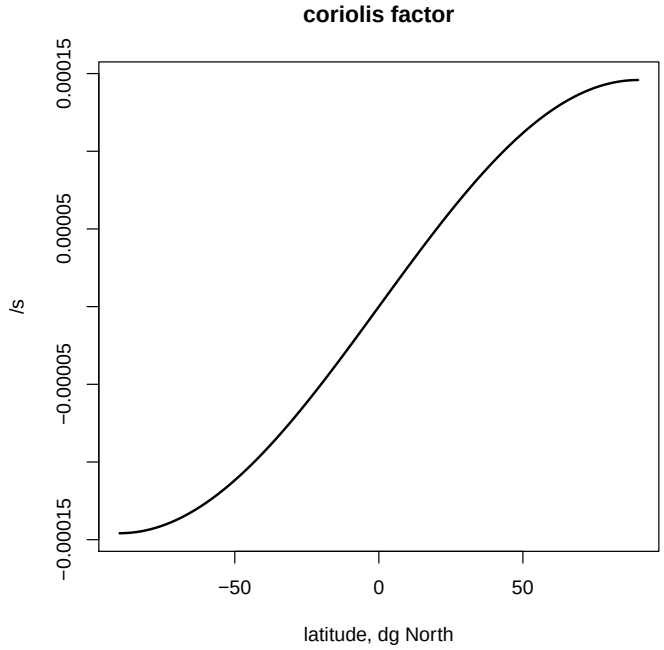

Figure 1: The coriolis function

```
3  gasCt1    0.08205784   L*atm/K/mol          ideal gas constant
4  gasCt2      8.314472   m3*Pa/K/mol          ideal gas constant
5  gasCt3       83.1451 cm3*bar/K/mol          ideal gas constant
6       F       96485.3         C/mol charge per mol of electrons
7      B1 1.3806504e-23           J/K          Boltzmann constant
8      B2  8.617343e-05           eV/K          Boltzmann constant
```

3. functions

3.1. coriolis

Estimates the coriolis factor, f, units sec^{-1} according to the formula: f=2*omega*sin(lat), where omega=7.292e-5 radians/sec

```
> plot(-90:90,coriolis(-90:90),xlab="latitude, dg North",
+                  ylab= "/s" , main ="coriolis factor",type="l",lwd=2)
```

3.2. heat capacity

Estimates the heat capacity of seawater, using the UNESCO 1983 polynomial (Fofonoff and Millard 1983)

```
> cp(S=40,T=1:20)

 [1] 3956.080 3955.898 3955.883 3956.021 3956.296 3956.697 3957.209 3957.819
 [9] 3958.516 3959.288 3960.124 3961.013 3961.945 3962.911 3963.900 3964.906
[17] 3965.918 3966.931 3967.936 3968.927
```

3.3. molecular diffusion coefficients

Calculates molecular and ionic diffusion coefficients (cm2/hour), for several species at given salinity (S) temperature (T) and pressure (P).

Based on the code "CANDI" by Bernie Boudreau (Boudreau 1996).

```
> diffcoeff(S=15,T=15)*24  # cm2/day

        O2       CO2       NH3       H2S       CH4       HCO3       CO3       NH4
1 1.429133 1.205394 1.422475 1.229416 1.132952 0.7692867 0.6126654 1.314530
        HS       NO3     H2PO4      HPO4       PO4         H       OH         Ca
1 1.214024 1.283122 0.6168532 0.4954089 0.399091 6.509832 3.54366 0.5263981
         Mg        Fe        Mn       SO4     H3PO4      BOH3      BOH4    H4SiO4
1 0.4681886 0.465676 0.4610695 0.700189 0.5558052 0.7601998 0.6651748 0.6881766

> diffcoeff(T=10)$O2

[1] 0.04930298

> difftemp <- diffcoeff(T=0:30)[,1:13]
> diffsal <- diffcoeff(S=0:35)[,1:13]

> matplot(0:30,difftemp,xlab="temperature",ylab="cm2/hour",
+         main="Molecular/ionic diffusion",type="l")
> legend("topleft",ncol=2,cex=0.8,title="mean",col=1:13,lty=1:13,
+         legend=cbind(names(difftemp),format(colMeans(difftemp),digits=4)))
```

3.4. shear viscosity of water

Calculates the shear viscosity of water, in centipoise. Valid for 0<T<30 and 0<S<36.

Based on the code "CANDI" by Bernie Boudreau (Boudreau 1996).

```
> plot(0:30,viscosity(S=35,T=0:30,P=1),xlab="temperature",ylab="centipoise",
+        main="shear viscosity of water",type="l")
>   lines(0:30,viscosity(S=0,T=0:30,P=1),col="red")
>   lines(0:30,viscosity(S=35,T=0:30,P=100),col="blue")
>   legend("topright",col=c("black","red","blue"),lty=1,
+         legend=c("S=35,P=1","S=0,P=1","S=35,P=100"))
```

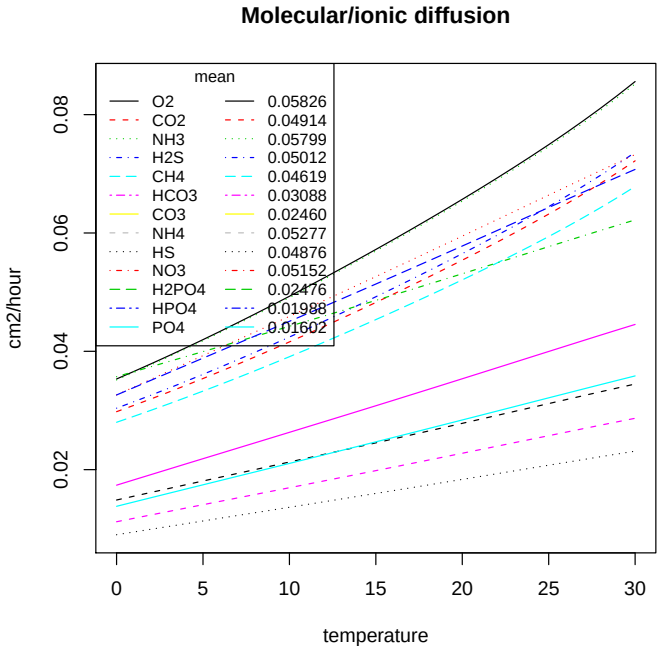

Figure 2: molecular diffusion coefficients as a function of temperature

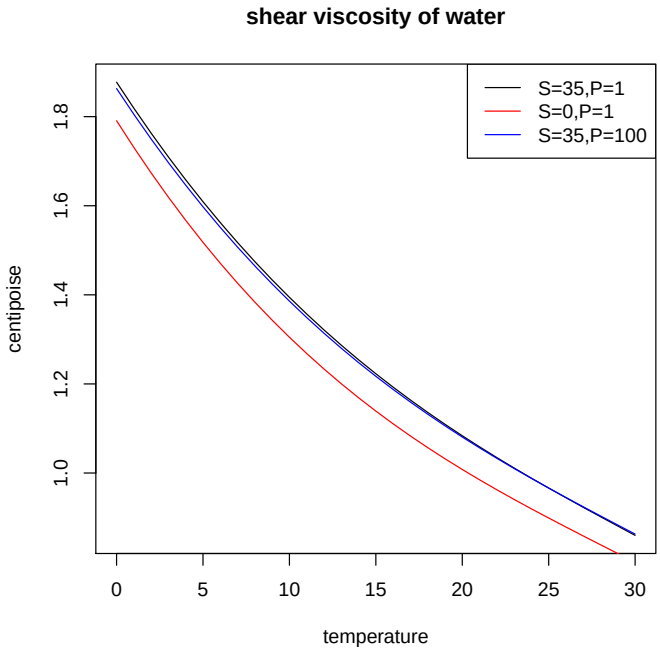

Figure 3: shear viscosity of water as a function of temperature

3.5. atmospheric composition

```
> atmComp("O2")

     O2
0.20946

> atmComp()

        He          Ne          N2          O2          Ar          Kr         CH4
5.2400e-06 1.8180e-05 7.8084e-01 2.0946e-01 9.3400e-03 1.1400e-06 1.7450e-06
       CO2         N2O          H2          Xe          CO          O3
3.6500e-04 3.1400e-07 5.5000e-07 8.7000e-08 5.0000e-08 1.0000e-08
```

3.6. saturated oxygen concentrations

```
> O2sat(T = 20)

[1] 9.067446
attr(,"unit")
[1] "(g/m3)"

> T <- seq(0, 30, 0.1)

>   plot(T, O2sat(T = T), type = "l", ylim = c(0, 15), lwd=2)
>   lines(T, O2sat(S = 0, T = T, method = "Weiss"), col = "yellow",
+         lwd = 2, lty = "dashed")
>   lines(T, O2sat(S = 35, T = T, method = "Weiss"), col = "red", lwd = 2)
```

3.7. solubilities and saturated concentrations

```
> satconc(gas="O2")

      O2
210.9798

> Temp<-seq(from=0,to=30,by=0.1)
> Sal <- seq(from=0,to=35,by=0.1)

> mf  <-par(mfrow=c(2,2))
> plot(Temp,solubility(T=Temp,gas="CCl4"),xlab="temperature",
+      ylab="mmol/m3/atm",main="solubility (S=35)",type="l",lwd=2,ylim=c(0,100000))
> lines(Temp,solubility(T=Temp,gas="CO2"),col="red",lwd=2)
```

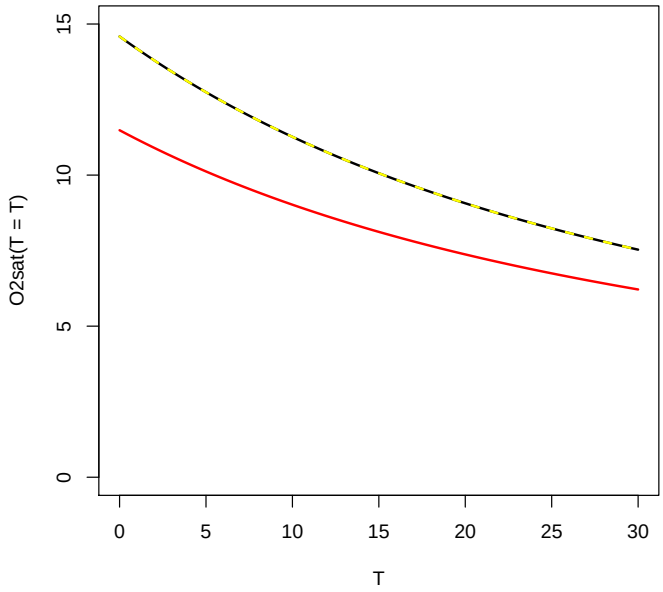

Figure 4: Oxygen saturated concentration as a function of temperature, and for different salinities

```
> lines(Temp,solubility(T=Temp,gas="N2O"),col="blue",lwd=2)
> lines(Temp,solubility(T=Temp,gas="Rn"),col="green",lwd=2)
> lines(Temp,solubility(T=Temp,gas="CCl2F2"),col="yellow",lwd=2)
> legend("topright",col=c("black","red","blue","green","yellow"),lwd=2,
+        legend=c("CCl4","CO2","N2O","Rn","CCl2F2"))
> plot(Temp,solubility(T=Temp,gas="Kr"),xlab="temperature",
+      ylab="mmol/m3/atm",main="solubility (S=35)",type="l",lwd=2,ylim=c(0,4000))
> lines(Temp,solubility(T=Temp,gas="CH4"),col="red",lwd=2)
> lines(Temp,solubility(T=Temp,gas="Ar"),col="blue",lwd=2)
> lines(Temp,solubility(T=Temp,gas="O2"),col="green",lwd=2)
> lines(Temp,solubility(T=Temp,gas="N2"),col="yellow",lwd=2)
> lines(Temp,solubility(T=Temp,gas="Ne"),col="grey",lwd=2)
> legend("topright",col=c("black","red","blue","green","yellow","grey"),lwd=2,
+        legend=c("Kr","CH4","Ar","O2","N2","Ne"))
> plot(Temp,satconc(T=Temp,gas="N2"),xlab="temperature",log="y",
+      ylab="mmol/m3",main="Saturated conc (S=35)",type="l",lwd=2,ylim=c(1,700))
> lines(Temp,satconc(T=Temp,gas="CO2"),col="red",lwd=2)
> lines(Temp,satconc(T=Temp,gas="O2"),col="blue",lwd=2)
> lines(Temp,satconc(T=Temp,gas="CH4"),col="green",lwd=2)
> lines(Temp,satconc(T=Temp,gas="N2O"),col="yellow",lwd=2)
> plot(Sal,satconc(S=Sal,gas="N2"),xlab="salinity",log="y",
+      ylab="mmol/m3",main="Saturated conc (T=20)",type="l",lwd=2,ylim=c(1e-3,700))
> lines(Sal,satconc(S=Sal,gas="CO2"),col="red",lwd=2)
```

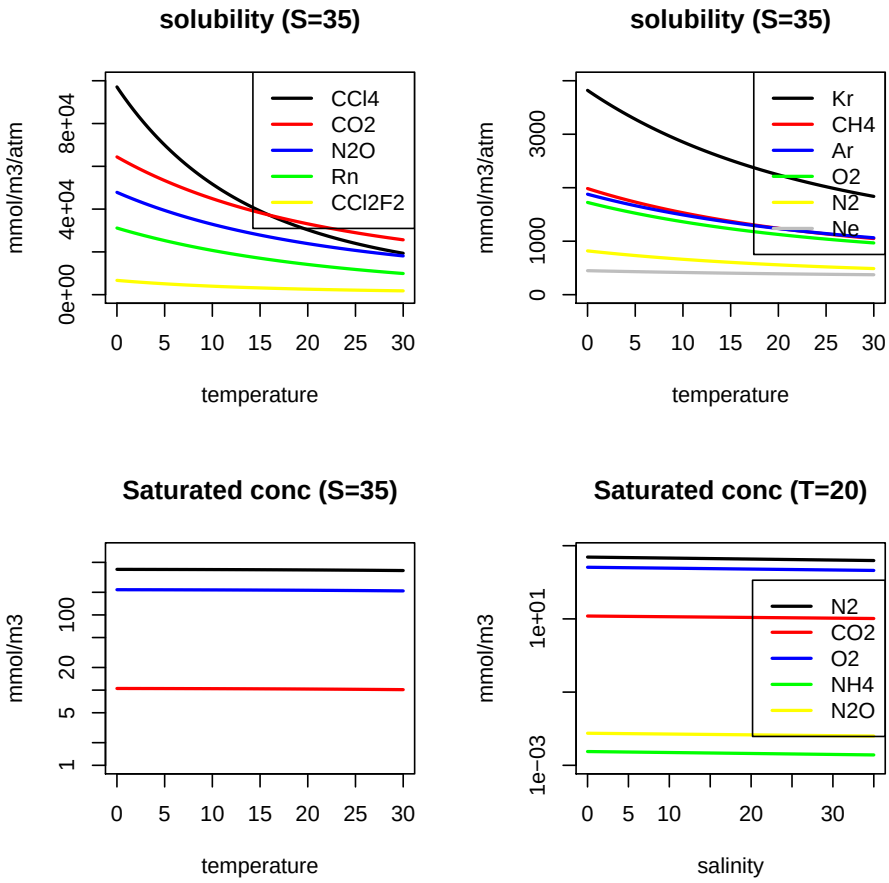

Figure 5: Saturated concentrations and solubility as a function of temperature and salinity, and for different species

```
> lines(Sal,satconc(S=Sal,gas="O2"),col="blue",lwd=2)
> lines(Sal,satconc(S=Sal,gas="CH4"),col="green",lwd=2)
> lines(Sal,satconc(S=Sal,gas="N2O"),col="yellow",lwd=2)
> legend("right",col=c("black","red","blue","green","yellow"),lwd=2,
+        legend=c("N2","CO2","O2","NH4","N2O"))
> par("mfrow"=mf)
```

3.8. Schmidt number and gas transfer velocity

```
> schmidt(gas="CO2",T=20)
```

```
[1] 665.988
```

```
> useq <- 0:15
```

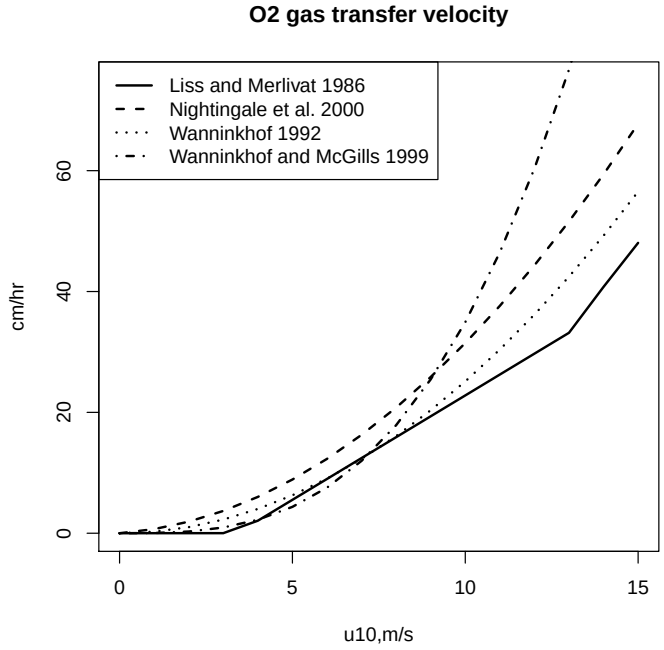

Figure 6: gas transfer velocity for seawater

```
> plot(useq,gastransfer(u10=useq),type="l",lwd=2,xlab="u10,m/s",
+      ylab="cm/hr", main="O2 gas transfer velocity",ylim=c(0,75))
> lines(useq,gastransfer(u10=useq,method="Nightingale"),lwd=2,lty=2)
> lines(useq,gastransfer(u10=useq,method="Wanninkhof1"),lwd=2,lty=3)
> lines(useq,gastransfer(u10=useq,method="Wanninkhof2"),lwd=2,lty=4)
> legend("topleft",lty=1:4,lwd=2,legend=c("Liss and Merlivat 1986",
+ "Nightingale et al. 2000","Wanninkhof 1992","Wanninkhof and McGills 1999"))
```

3.9. Concentration of conservative species in seawater

```
> salconc(S=seq(0,35,by=5))
```

```
     Borate   Calcite  Sulphate  Fluoride
1   0.00000     0.000     0.000  0.000000
2  59.42857  1468.571  4033.633  9.760629
3 118.85714  2937.143  8067.267 19.521257
4 178.28571  4405.714 12100.900 29.281886
5 237.71429  5874.286 16134.534 39.042515
6 297.14286  7342.857 20168.167 48.803144
7 356.57143  8811.429 24201.801 58.563772
8 416.00000 10280.000 28235.434 68.324401
```

4. conversions

4.1. gram, mol, liter conversions

gram to moles and vice versa

```
> g2mol("CO3")

       CO3
0.01666419

> g2mol("HCO3")

      HCO3
0.01638892

> g2mol(c("C2H5OH", "CO2", "H2O"))

    C2H5OH        CO2        H2O
0.02170683 0.02272237 0.05550844

> mol.weight(c("SiOH4", "NaHCO3", "C6H12O6", "Ca(HCO3)2", "Pb(NO3)2", "(NH4)2SO4"))

    SiOH4    NaHCO3   C6H12O6 Ca(HCO3)2  Pb(NO3)2 (NH4)2SO4
 48.11666  84.00661 180.15588 162.11168 331.20980 132.13952
```

liter to moles and vice versa

```
> l2mol(gas="O2",T=0)*1000

[1] 44.67377

> l2mol(x=1:6,T=0)

[1] 0.04461469 0.08922938 0.13384407 0.17845875 0.22307344 0.26768813

> mol2l(T=1:10,gas="O2")

 [1] 22.46678 22.54906 22.63134 22.71362 22.79590 22.87817 22.96045 23.04272
 [9] 23.12499 23.20726
```

molar volume of an ideal gas

```
> mol.vol()

[1] 24.46559
```

```
> mol.vol(T=1:10)

 [1] 22.49620 22.57826 22.66032 22.74237 22.82443 22.90649 22.98855 23.07061
 [9] 23.15266 23.23472
```

4.2. pressure conversions

```
> baroconv(1, "atm")

        Pa      bar       at atm     torr
1 101325.3 1.013253 1.033214   1 760.0008
```

4.3. salinity and chlorinity

```
> sal2cl(S=35)

[1] 19.37394
```

5. finally

This vignette is mainly a Sweave (Leisch 2002) translation of part of the **marelac** help files.

References

Boudreau B (1996). "A method-of-lines code for carbon and nutrient diagenesis in aquatic sediments." *Computers and Geosciences*, **22 (5)**, 479–496.

Fofonoff P, Millard RJ (1983). *Algorithms for computation of fundamental properties of seawater.*, volume 44. Unesco. 53 pp.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), "Compstat 2002 - Proceedings in Computational Statistics," pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL http://www.stat.uni-muenchen.de/~leisch/Sweave.

Soetaert K, Petzoldt T, Meysman F (2008). *marelac: Constants, conversion factors, utilities for the MArine, Riverine, Estuarine, LAcustrine and Coastal sciences*. R package version 1.4.

Affiliation:

Karline Soetaert
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO)
4401 NT Yerseke, Netherlands
E-mail: k.soetaert@nioo.knaw.nl
URL: http://www.nioo.knaw.nl/ppages/ksoetaert

Thomas Petzoldt
Institut für Hydrobiologie
Technische Universität Dresden
01062 Dresden, Germany
E-mail: thomas.petzoldt@tu-dresden.de
URL: http://tu-dresden.de/Members/thomas.petzoldt/

Filip Meysman
Laboratory of Analytical and Environmental Chemistry
Vrije Universiteit Brussel (VUB)
Pleinlaan 2
1050 Brussel, Belgium.
E-mail: filip.meysman@vub.ac.be