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
 1.007940   4.002602   6.941000   9.012182  10.811000  12.010700
        N          O          F         Ne         Na         Mg
14.006700  15.999400  18.998403  20.179700  22.989769  24.305000
       Al         Si          P          S         Cl         Ar
```

```
 26.981539  28.085500  30.973762  32.065000  35.453000  39.948000
         K         Ca         Sc         Ti          V         Cr
 39.098300  40.078000  44.955912  47.867000  50.941500  51.996100
        Mn         Fe         Co         Ni         Cu         Zn
 54.938045  55.845000  58.933195  58.693400  63.546000  65.409000
        Ga         Ge         As         Se         Br         Kr
 69.723000  72.640000  74.921600  78.960000  79.904000  83.798000
        Rb         Sr          Y         Zr         Nb         Mo
 85.467800  87.620000  88.905850  91.224000  92.906380  95.940000
        Tc         Ru         Rh         Pd         Ag         Cd
        NA 101.070000 102.905500 106.420000 107.868200 112.411000
        In         Sn         Sb         Te          I         Xe
114.818000 118.710000 121.760000 127.600000 126.904470 131.293000
        Cs         Ba         La         Ce         Pr         Nd
132.905452 137.327000 138.905470 140.116000 140.907650 144.242000
        Pm         Sm         Eu         Gd         Tb         Dy
        NA 150.360000 151.964000 157.250000 158.925350 162.500000
        Ho         Er         Tm         Yb         Lu         Hf
164.930320 167.259000 168.934210 173.040000 174.967000 178.490000
        Ta          W         Re         Os         Ir         Pt
180.947880 183.840000 186.207000 190.230000 192.217000 195.084000
        Au         Hg         Tl         Pb         Bi         Po
196.966569 200.590000 204.383300 207.200000 208.980400         NA
        At         Rn         Fr         Ra         Ac         Th
        NA         NA         NA         NA         NA 232.038060
        Pa          U         Np         Pu         Am         Cm
231.035880 238.028910         NA         NA         NA         NA
        Bk         Cf         Es         Fm         Md         No
        NA         NA         NA         NA         NA         NA
        Lr         Rf         Db         Sg         Bh         Hs
        NA         NA         NA         NA         NA         NA
        Mt         Ds         Rg
        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")))))
```

Figure 1: The coriolis function

	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
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	P0	101325	Pa	one standard atmosphere
8	B1	1.3806504e-23	J/K	Boltzmann constant
9	B2	8.617343e-05	eV/K	Boltzmann constant

3. physical 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. 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
1 1.429209 1.205459 1.422551 1.229482 1.133013 0.7693278 0.6126982
       NH4       HS      NO3     H2PO4      HPO4       PO4       H
1 1.314600 1.214089 1.283190 0.6168861 0.4954354 0.3991123 6.51018
        OH        Ca        Mg        Fe        Mn       SO4    H3PO4
1 3.543850 0.5264263 0.4682136 0.4657009 0.4610941 0.7002265 0.555835
       BOH3      BOH4    H4SiO4
1 0.7602404 0.6652104 0.6882134
```

```
> diffcoeff(t=10)$O2
```

```
[1] 0.04930629
```

```
> 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)))
```

3.3. 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"))
```

3.4. atmospheric composition

```
> atmComp("O2")
```

Figure 2: molecular diffusion coefficients as a function of temperature

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

```
     O2
0.20946
```

```
> atmComp()
```

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

```
> sum(atmComp())    #!
```

```
[1] 1.000032
```

4. dissolved gasses

4.1. saturated oxygen concentrations

```
> gas_O2sat(t = 20)
```

```
[1] 7.374404
attr(,"unit")
[1] "(g/m3)"
```

```
> t <- seq(0, 30, 0.1)
```

```
>   plot(t, gas_O2sat(t = t), type = "l", ylim = c(0, 15), lwd=2)
>   lines(t, gas_O2sat(S = 0, t = t, method = "Weiss"), col = "yellow",
+         lwd = 2, lty = "dashed")
>   lines(t, gas_O2sat(S = 35, t = t, method = "Weiss"), col = "red", lwd = 2)
```

4.2. solubilities and saturated concentrations

```
> gas_satconc(x="O2")
```

```
           O2
[1,] 213.7760
```

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

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

```
> #
> mf  <-par(mfrow=c(2,2))
> #
> gs  <-gas_solubility(t=Temp)
> x   <- c("CCl4","CO2","N2O","Rn","CCl2F2")
> matplot(Temp,gs[,x],type="l",lty=1,lwd=2,xlab="temperature",
+       ylab="mmol/m3",main="solubility (S=35)")
> legend("topright",col=1:5,lwd=2,legend=x)
> #
> x2 <- c("Kr","CH4","Ar","O2","N2","Ne")
> matplot(Temp,gs[,x2],type="l",lty=1,lwd=2,xlab="temperature",
+       ylab="mmol/m3",main="solubility (S=35)")
> legend("topright",col=1:6,lwd=2,legend=x2)
> #
>
> x <- c("N2","CO2","O2","CH4","N2O")
> gsat  <-gas_satconc(t=Temp,x=x)
> matplot(Temp,gsat,type="l",xlab="temperature",log="y", lty=1,
+       ylab="mmol/m3",main="Saturated conc (S=35)",lwd=2)
> legend("right",col=1:5,lwd=2,legend=x)
> #
> gsat  <-gas_satconc(S=Sal,x=x)
> matplot(Sal,gsat,type="l",xlab="salinity",log="y", lty=1,
```

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

```
+      ylab="mmol/m3",main="Saturated conc (T=20)",lwd=2)
> legend("right",col=1:5,lwd=2,legend=x)
> #
> par("mfrow"=mf)
```

4.3. Schmidt number and gas transfer velocity

```
> gas_schmidt(x="CO2",t=20)
```

```
[1] 665.988
```

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

```
> plot(useq,gas_transfer(u10=useq,x="O2"),type="l",lwd=2,xlab="u10,m/s",
+      ylab="cm/hr", main="O2 gas transfer velocity",ylim=c(0,75))
```

Figure 6: gas transfer velocity for seawater

```
> lines(useq,gas_transfer(u10=useq,x="O2",method="Nightingale"),lwd=2,lty=2)
> lines(useq,gas_transfer(u10=useq,x="O2",method="Wanninkhof1"),lwd=2,lty=3)
> lines(useq,gas_transfer(u10=useq,x="O2",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"))
```

5. seawater properties

5.1. Concentration of conservative species in seawater

```
> sw_conserv(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
```

5.2. seawater properties

Several sewater profperties, using (R 2008) or according to the UNESCO 1983 polynomial (Fofonoff and Millard 1983) are included. For instance, the heat capacity:

```
> sw_cp(S=40,t=1:20)

 [1] 3958.545 3959.028 3959.576 3960.180 3960.831 3961.523 3962.247
 [8] 3962.997 3963.768 3964.553 3965.348 3966.148 3966.949 3967.747
[15] 3968.540 3969.324 3970.098 3970.859 3971.605 3972.336

> sw_cp(S=40,t=1:20,UNESCO=TRUE)

 [1] 3956.080 3955.898 3955.883 3956.021 3956.296 3956.697 3957.209
 [8] 3957.819 3958.516 3959.288 3960.124 3961.013 3961.945 3962.911
[15] 3963.900 3964.906 3965.918 3966.931 3967.936 3968.927
```

The dependency on salinity, and temperature for some of those is hereby plotted, using a function

```
> plotST <- function(fun,title)
+ {
+   Sal <-  seq(0,40,by=0.5)
+   Temp<- seq(-5,40,by=0.5)
+
+   Val <- outer(X=Sal,Y=Temp,FUN=function(X,Y) fun(S=X, t=Y))
+   contour(Sal,Temp,Val,xlab="Salinity",ylab="temperature",
+           main=title,nlevel=20)
+ }
> par (mfrow=c(3,2))
> par(mar=c(4,4,3,2))
> plotST(sw_cp,"Heat capacity")
> plotST(sw_entropy,"Entropy")
> plotST(sw_enthalpy,"Enthalpy")
> plotST(sw_dens,"density")
> plotST(sw_svel,"sound velocity")
> plotST(sw_kappa,"isentropic compressibility")
```

6. conversions

6.1. gram, mol, liter conversions

gram to moles and vice versa

```
> 1/mol.weight("CO3")
```

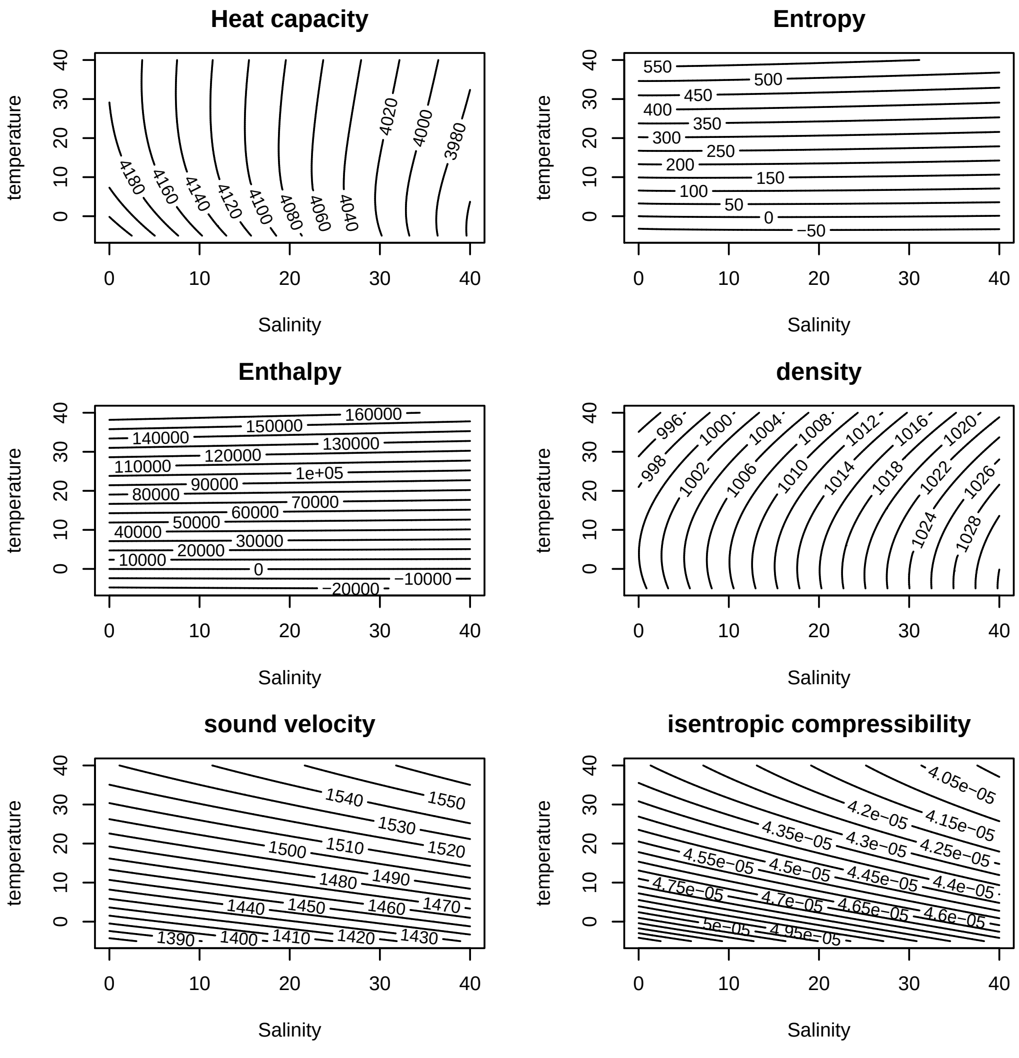

Figure 7: seawater properties as a function of salinity and temperture - see text for R-code

```
       CO3
0.01666419
```

```
> 1/mol.weight("HCO3")
```

```
      HCO3
0.01638892
```

```
> 1/mol.weight(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

```
> 1/mol.vol(x="O2",t=0)*1000
```

```
      O2
45.86736
```

```
> 1/mol.vol(x="O2",q=1:6,t=0)
```

```
              O2
[1,] 0.045867360
[2,] 0.022933679
[3,] 0.015289117
[4,] 0.011466842
[5,] 0.009173472
[6,] 0.007644560
```

```
> 1/mol.vol(t=1:10,x="O2")
```

```
              O2
 [1,] 0.04569936
 [2,] 0.04553258
 [3,] 0.04536702
 [4,] 0.04520267
 [5,] 0.04503950
 [6,] 0.04487752
 [7,] 0.04471669
 [8,] 0.04455702
 [9,] 0.04439848
[10,] 0.04424108
```

molar volume of an ideal gas

```
> mol.vol(x="ideal")

   ideal
24.46559

> mol.vol(x="ideal",t=1:10)

         ideal
 [1,] 22.49620
 [2,] 22.57826
 [3,] 22.66032
 [4,] 22.74237
 [5,] 22.82443
 [6,] 22.90649
 [7,] 22.98855
 [8,] 23.07061
 [9,] 23.15266
[10,] 23.23472
```

6.2. pressure conversions

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

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

6.3. salinity and chlorinity

```
> convert_StoCl(S=35)

[1] 19.37394
```

7. 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.

Feistel R (2008). "A Gibbs function for seawater thermodynamics for -6 to 80 dgC and salinity up to 120 g/kg." *Deep-Sea Res*, **I, 55**, 1639–1671.

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