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; conversions between different barometric units, temperature units, salinity units.
- physical functions, e.g. to estimate concentrations of conservative substances as a function of salinity, gas transfer coefficients, diffusion coefficients, estimating the coriolis force, gravity ...
- the thermophysical properties of the seawater, as from the UNESCO polynomial (P. and Millard 1983) or as from the more recent derivation based on a gibbs funtion (Feistel 2008), (T.J., Feistel, FJ, Jackett, Wright, King, Marion, A., and P 2009).

2. Constants

2.1. Physical constants

Dataset `Constants` contains commonly used physical and chemical constants, as in (P.J. and Taylor 2005):

```
> 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
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	E	1.60217653e-19	C	Elementary charge
7	F	96485.3	C/mol	charge per mol of electrons
8	P0	101325	Pa	one standard atmosphere
9	B1	1.3806505e-23	J/K	Boltzmann constant
10	B2	8.617343e-05	eV/K	Boltzmann constant
11	Na	6.0221415e+23	mol-1	Avogadro constant
12	C	299792458	m s-1	Vacuum light speed

2.2. Ocean characteristics

Dataset `Oceans` contains surfaces and volumes of the world ocean as in (Sarmiento and Gruber 2006):

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

	acronym	value	units	description
1	Mass	1.35e+25	kg	total mass of the oceans
2	Vol	1.34e+18	kg	total volume of the oceans
3	VolSurf	1.81e+16	kg	volume of the surface ocean (0-50m)
4	VolDeep	9.44e+17	kg	volume of the deep ocean (>1200m)
5	Area	3.58e+14	m2	total area of the oceans
6	AreaIF	3.32e+14	m2	annual mean ice-free area of the oceans
7	AreaAtl	7.5e+13	m2	area of the Atlantic ocean, >45dgS
8	AreaPac	1.51e+14	m2	area of the Pacific ocean, >45dgS
9	AreaInd	5.7e+13	m2	area of the Indian ocean, >45dgS
10	AreaArct	9.6e+12	m2	area of the Arctic ocean
11	AreaEncl	4.5e+12	m2	area of enclosed seas (e.g. Mediterranean)
12	Depth	3690	m	mean depth of the oceans
13	RiverFlow	3.7e+13	m3/yr	Total river flow

2.3. AtomicWeight

Dataset `AtomicWeight` holds the atomic weight of various chemical elements, as in (M.E. 2006); The data set contains `NA` for elements which have no stable isotopes (except U, Th,

Pa). The data set can be called in two versions. `AtomicWeight` shows the full table and `atomicweight` can be used for symbolic computations with the elements.

```
> AtomicWeight
```

	Number	Name	Symbol	Weight	Footnotes
1	1	hydrogen	H	1.00794(7)	gmr
2	2	helium	He	4.002602(2)	gr
3	3	lithium	Li	6.941(2)	+gmr
4	4	beryllium	Be	9.012182(3)	
5	5	boron	B	10.811(7)	gmr
6	6	carbon	C	12.0107(8)	gr
7	7	nitrogen	N	14.0067(2)	gr
8	8	oxygen	O	15.9994(3)	gr
9	9	fluorine	F	18.9984032(5)	
10	10	neon	Ne	20.1797(6)	gm
11	11	sodium	Na	22.98976928(2)	
12	12	magnesium	Mg	24.3050(6)	
13	13	aluminium	Al	26.9815386(8)	
14	14	silicon	Si	28.0855(3)	r
15	15	phosphorus	P	30.973762(2)	
16	16	sulfur	S	32.065(5)	gr
17	17	chlorine	Cl	35.453(2)	gmr
18	18	argon	Ar	39.948(1)	gr
19	19	potassium	K	39.0983(1)	
20	20	calcium	Ca	40.078(4)	g
21	21	scandium	Sc	44.955912(6)	
22	22	titanium	Ti	47.867(1)	
23	23	vanadium	V	50.9415(1)	
24	24	chromium	Cr	51.9961(6)	
25	25	manganese	Mn	54.938045(5)	
26	26	iron	Fe	55.845(2)	
27	27	cobalt	Co	58.933195(5)	
28	28	nickel	Ni	58.6934(2)	
29	29	copper	Cu	63.546(3)	r
30	30	zinc	Zn	65.409(4)	
31	31	gallium	Ga	69.723(1)	
32	32	germanium	Ge	72.64(1)	
33	33	arsenic	As	74.92160(2)	
34	34	selenium	Se	78.96(3)	r
35	35	bromine	Br	79.904(1)	
36	36	krypton	Kr	83.798(2)	gm
37	37	rubidium	Rb	85.4678(3)	g
38	38	strontium	Sr	87.62(1)	gr
39	39	yttrium	Y	88.90585(2)	
40	40	zirconium	Zr	91.224(2)	g
41	41	niobium	Nb	92.90638(2)	

42	42	molybdenum	Mo	95.94(2)	g
43	43	technetium	Tc		*
44	44	ruthenium	Ru	101.07(2)	g
45	45	rhodium	Rh	102.90550(2)	
46	46	palladium	Pd	106.42(1)	g
47	47	silver	Ag	107.8682(2)	g
48	48	cadmium	Cd	112.411(8)	g
49	49	indium	In	114.818(3)	
50	50	tin	Sn	118.710(7)	g
51	51	antimony	Sb	121.760(1)	g
52	52	tellurium	Te	127.60(3)	g
53	53	iodine	I	126.90447(3)	
54	54	xenon	Xe	131.293(6)	gm
55	55	caesium	Cs	132.9054519(2)	
56	56	barium	Ba	137.327(7)	
57	57	lanthanum	La	138.90547(7)	g
58	58	cerium	Ce	140.116(1)	g
59	59	praseodymium	Pr	140.90765(2)	
60	60	neodymium	Nd	144.242(3)	g
61	61	promethium	Pm		*
62	62	samarium	Sm	150.36(2)	g
63	63	europium	Eu	151.964(1)	g
64	64	gadolinium	Gd	157.25(3)	g
65	65	terbium	Tb	158.92535(2)	
66	66	dysprosium	Dy	162.500(1)	g
67	67	holmium	Ho	164.93032(2)	
68	68	erbium	Er	167.259(3)	g
69	69	thulium	Tm	168.93421(2)	
70	70	ytterbium	Yb	173.04(3)	g
71	71	lutetium	Lu	174.967(1)	g
72	72	hafnium	Hf	178.49(2)	
73	73	tantalum	Ta	180.94788(2)	
74	74	tungsten	W	183.84(1)	
75	75	rhenium	Re	186.207(1)	
76	76	osmium	Os	190.23(3)	g
77	77	iridium	Ir	192.217(3)	
78	78	platinum	Pt	195.084(9)	
79	79	gold	Au	196.966569(4)	
80	80	mercury	Hg	200.59(2)	
81	81	thallium	Tl	204.3833(2)	
82	82	lead	Pb	207.2(1)	gr
83	83	bismuth	Bi	208.98040(1)	
84	84	polonium	Po		*
85	85	astatine	At		*
86	86	radon	Rn		*
87	87	francium	Fr		*
88	88	radium	Ra		*

```
89      89      actinium     Ac                      *
90      90       thorium     Th   232.03806(2)       *g
91      91  protactinium     Pa   231.03588(2)        *
92      92       uranium      U   238.02891(3)      *gm
93      93     neptunium     Np                      *
94      94     plutonium     Pu                      *
95      95     americium     Am                      *
96      96        curium     Cm                      *
97      97     berkelium     Bk                      *
98      98   californium     Cf                      *
99      99   einsteinium     Es                      *
100    100       fermium     Fm                      *
101    101   mendelevium     Md                      *
102    102      nobelium     No                      *
103    103    lawrencium     Lr                      *
104    104 rutherfordium     Rf                      *
105    105       dubnium     Db                      *
106    106    seaborgium     Sg                      *
107    107       bohrium     Bh                      *
108    108       hassium     Hs                      *
109    109    meitnerium     Mt                      *
110    110  darmstadtium     Ds                      *
111    111   roentgenium     Rg                      *
```

```
> AtomicWeight[8,]
```

```
  Number   Name Symbol     Weight Footnotes
8      8 oxygen      O 15.9994(3)        gr
```

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

```
[1] 18.01528
```

2.4. atmospheric composition

The atmospheric composition, given in units of the moles of each gas to the total of moles of gas in dry air is in function `atmComp`:

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

```
     O2
0.20946
```

```
> atmComp()
```

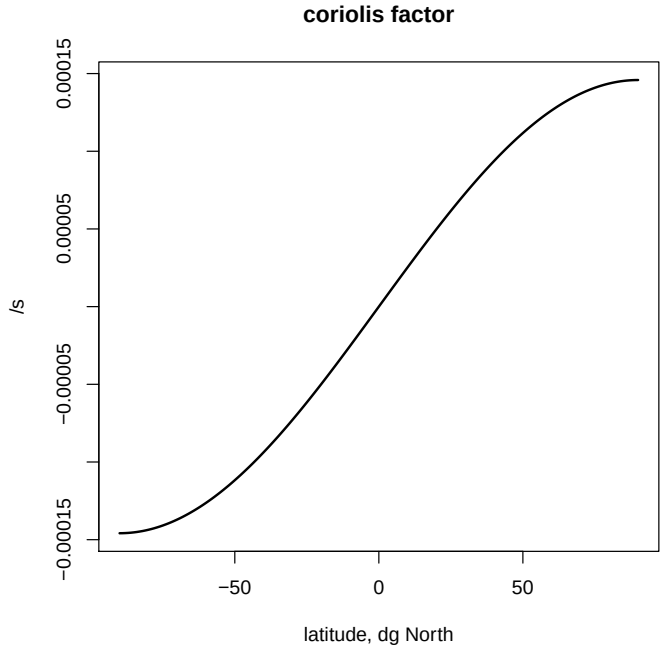

Figure 1: The coriolis function

```
        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
```

3. physical functions

3.1. coriolis

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

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

3.2. molecular diffusion coefficients

In function `diffcoeff` the molecular and ionic diffusion coefficients ($cm^2 hour^{-1}$), for several species at given salinity (S) temperature (t) and pressure (P) is estimated. The implementation is 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

`viscosity` calculates the shear viscosity of water, in centipoise ($0.01 gcm^{-1}sec^{-1}$). Valid for 0<t<30 and 0<S<36.

```
> 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"))
```

4. dissolved gasses

4.1. saturated oxygen concentrations

`gas_O2sat` estimates the saturated concentration of oxygen:

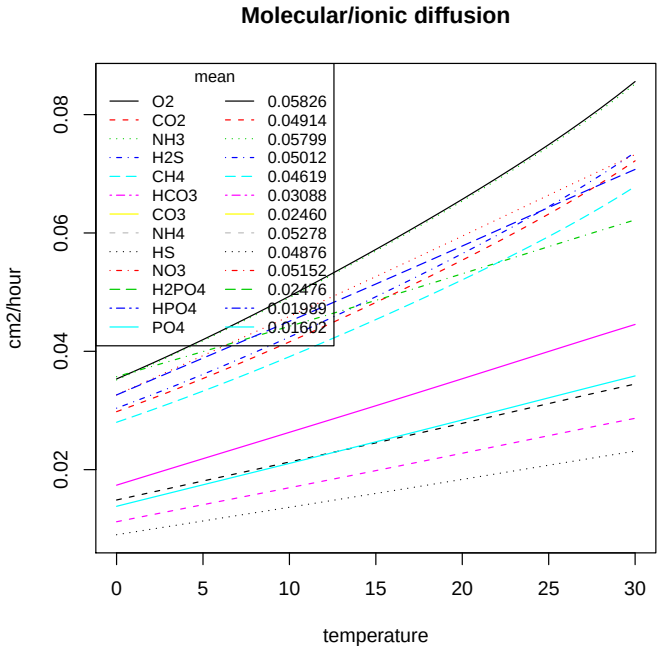

Figure 2: molecular diffusion coefficients as a function of temperature

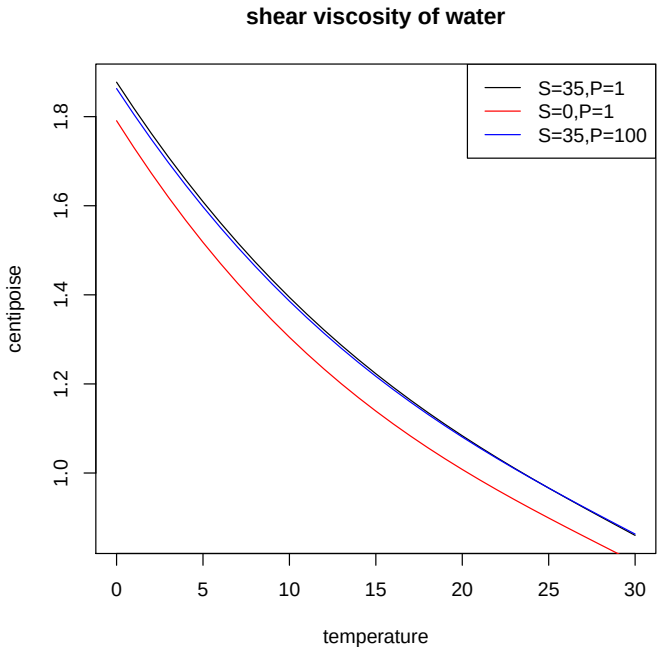

Figure 3: shear viscosity of water as a function of temperature

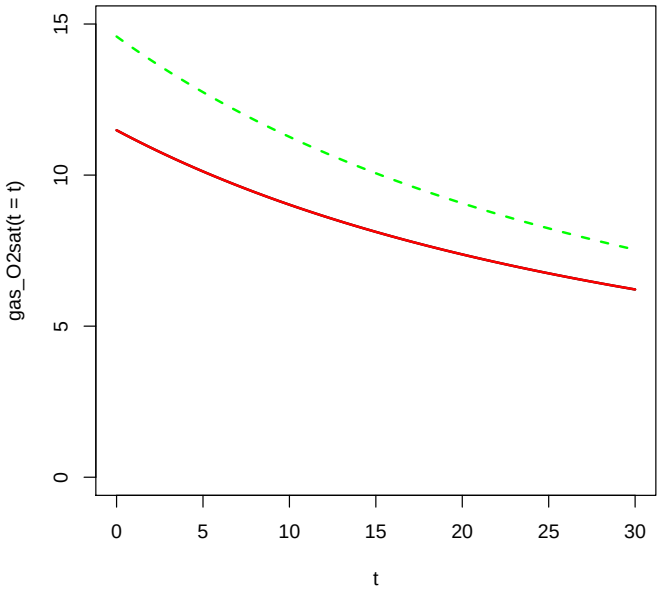

Figure 4: Oxygen saturated concentration as a function of temperature, and for different salinities

```
> gas_O2sat(t = 20)

[1] 7.374404

> 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 = "green",
+         lwd = 2, lty = "dashed")
>   lines(t, gas_O2sat(S = 35, t = t, method = "Weiss"), col = "red", lwd = 2)
```

4.2. solubilities and saturated concentrations

More solubilities and saturated concentrations are in functions `gas_solubility` and `gas_satconc`.

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

           O2
[1,] 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))
> #
> 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,
+      ylab="mmol/m3",main="Saturated conc (T=20)",lwd=2)
> legend("right",col=1:5,lwd=2,legend=x)
> #
> par("mfrow"=mf)
```

4.3. partial pressure of water vapor

`vapor` estimates the partial pessure of water vapor, divided by the atmospheric pressure.

```
> plot(0:30, vapor(t = 0:30), xlab = "Temperature, dgC", ylab = "pH2O/P",
+      type ="l")
```

4.4. Schmidt number and gas transfer velocity

The Schmidt number of a gas (`gas_schmidt`) is an essential quantity in the gas transfer velocity calculation (`gas_transfer`). The latter also depends on wind velocity (as measured 10 metres above sealevel).

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

[1] 665.988

> useq <- 0:15
```

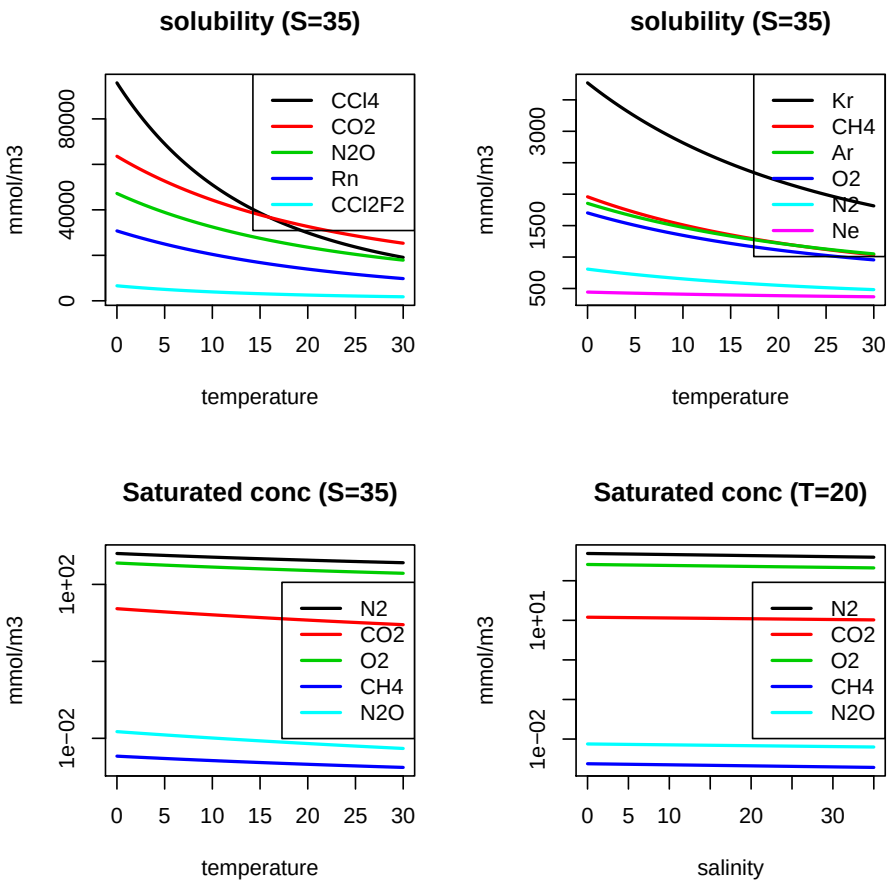

Figure 5: Saturated concentrations and solubility as a function of temperature and salinity, and for different species

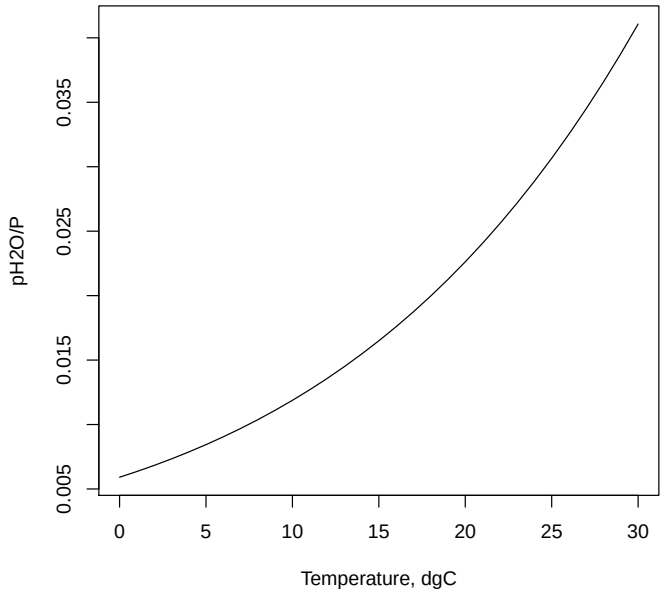

Figure 6: partial pressure of water in saturated air as a function of temperature

```
> 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))
> 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

Borate, calcite, sulphate and fluoride concentrations can be estimated as a function of the seawater salinity:

```
> 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
```

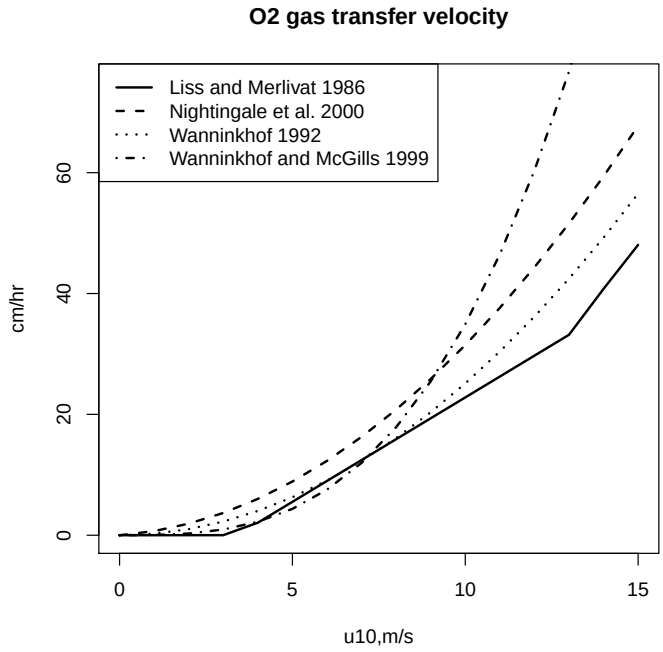

Figure 7: gas transfer velocity for seawater

```
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. Thermophysical seawater properties

marelac also implements several thermodynamic properties of seawater. Either one can choose the formulation based on the UNESCO polynomial (P. and Millard 1983), which has served the oceanographic community for decades, or the more recent derivation as in (Feistel 2008). In the latter case the estimates are based on three individual thermodynamic potentials for fluid water, for ice and for the saline contribution of seawater (the Helmholtz function for pure water, an equation of state for salt-free ice, in the form of a Gibbs potential function, and the saline part of the Gibbs potential).

Note that the formulations use a new salinity scale, termed "Reference composition salinity" (F.J., R., D.G., and T.J. 2008).

Below we plot all implemented thermophysical properties as a function of salinity and temperature.

```
> 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_gibbs,"Gibbs function")
> plotST(sw_cp,"Heat capacity")
> plotST(sw_entropy,"Entropy")
> plotST(sw_enthalpy,"Enthalpy")
> plotST(sw_dens,"Density")
> plotST(sw_svel,"Sound velocity")


> par (mfrow=c(3,2))
> par(mar=c(4,4,3,2))
> plotST(sw_kappa,"Isentropic compressibility")
> plotST(sw_kappa_t,"Isothermal compressibility")
> plotST(sw_alpha,"Thermal expansion coefficient")
> plotST(sw_beta,"Haline contraction coefficient")
> plotST(sw_adtgrad,"Adiabatic temperature gradient")
> par (mfrow=c(1,1))
```

6. conversions

Finally, several functions are included to convert between units of certain properties.

6.1. gram, mol, liter conversions

marelac function `molweight` converts from gram to moles and vice versa

```
> 1/molweight("CO3")
```

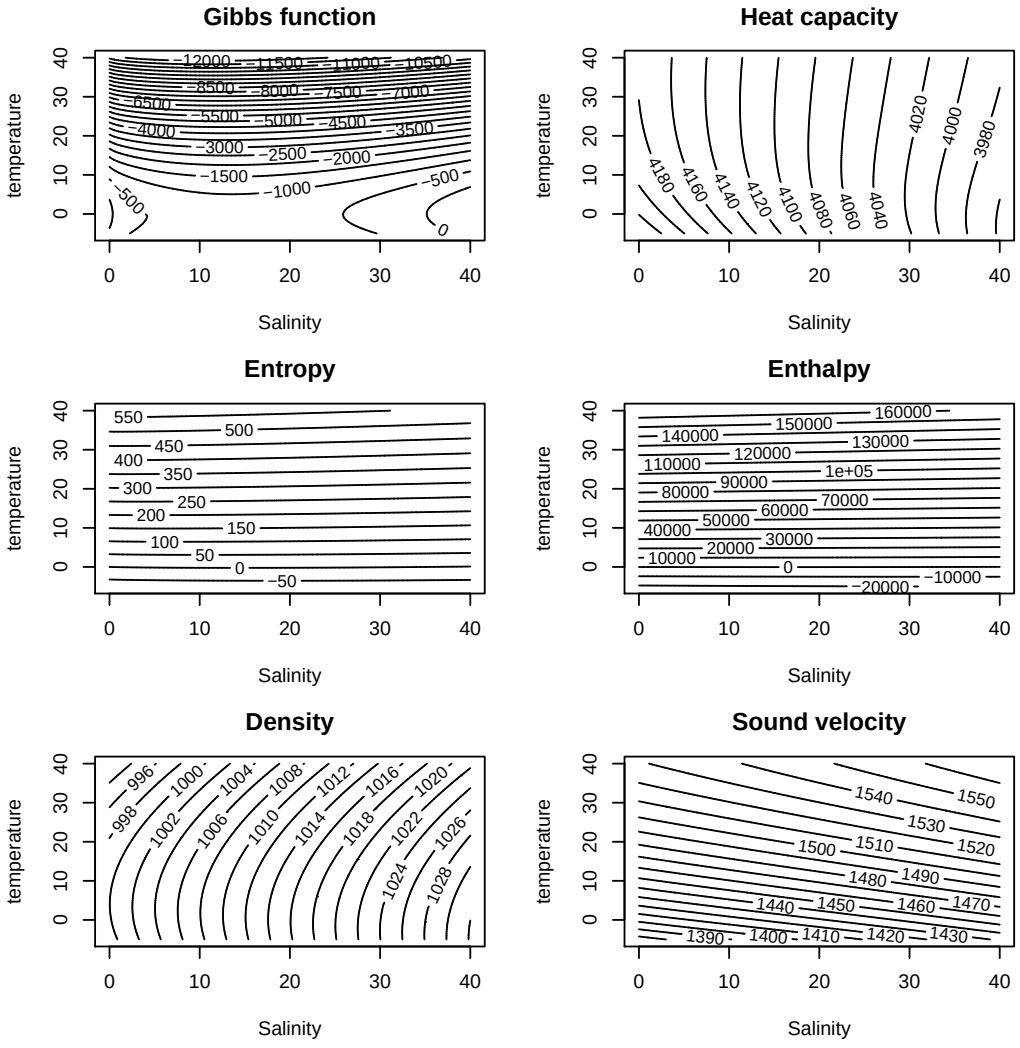

Figure 8: seawater properties as a function of salinity and temperature - see text for R-code

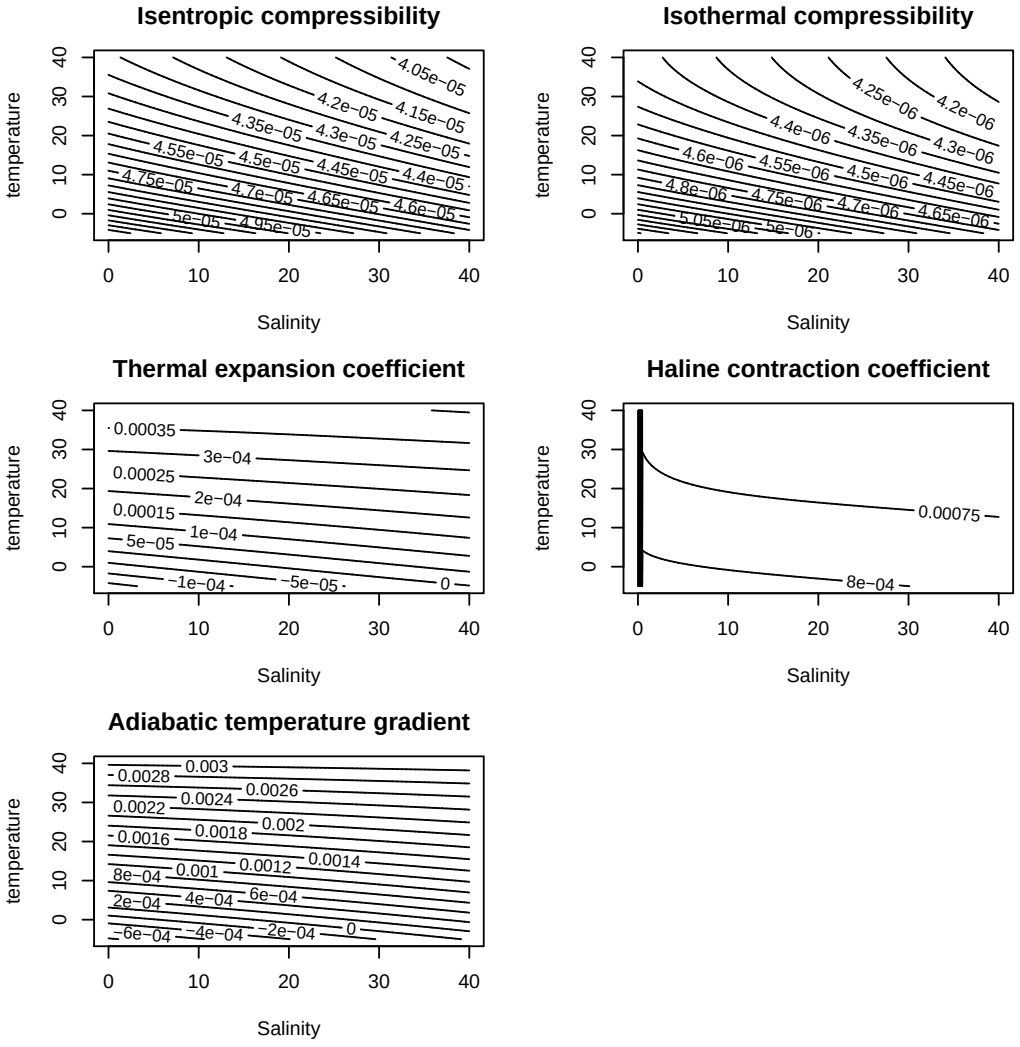

Figure 9: seawater properties as a function of salinity and temperature - continued - see text for R-code

```
       CO3
0.01666419

> 1/molweight("HCO3")

      HCO3
0.01638892

> 1/molweight(c("C2H5OH", "CO2", "H2O"))

    C2H5OH        CO2        H2O
0.02170683 0.02272237 0.05550844

> molweight(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
```

We can use that to estimate the importance of molecular weight on certain physical properties:

```
> gs <- gas_solubility()
> x <- colnames(gs)
> mw <- molweight(x)

> plot(mw,gs,type="n",xlab="molecular weight", ylab="solubility", log="y")
> text(mw,gs,x)
```

`molvol` estimates the volume of one liter of a gas. molar volume of an ideal gas

```
> molvol(x="ideal")

   ideal
24.46559

> molvol(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
```

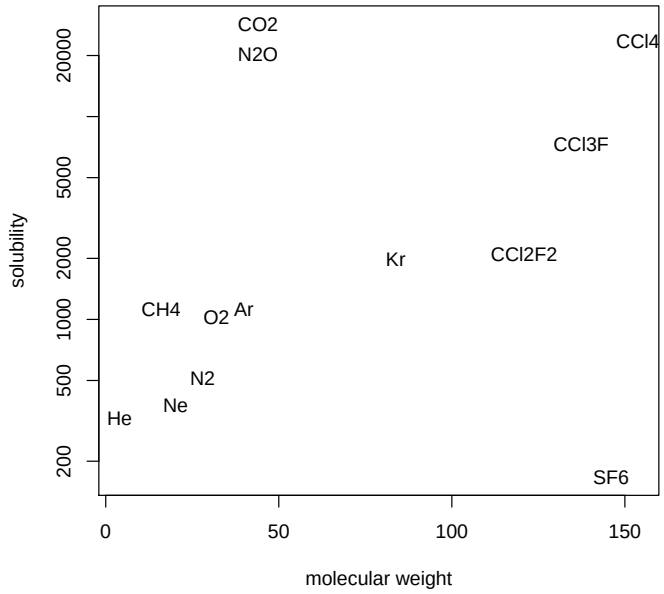

Figure 10: Gas solubility as a function of molecular weight see text for R-code

```
> 1/molvol(x="O2",t=0)*1000

      O2
45.86736

> 1/molvol(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/molvol(t=1:5,x=c("CO2","O2","N2O"))

            CO2         O2        N2O
[1,] 0.04588868 0.04569936 0.04590370
[2,] 0.04571975 0.04553258 0.04573458
[3,] 0.04555206 0.04536702 0.04556672
[4,] 0.04538560 0.04520267 0.04540009
[5,] 0.04522037 0.04503950 0.04523469
```

6.2. pressure conversions

`convert_p` converts between the different barometric scales:

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

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

6.3. temperature conversions

while `convert_T` converts between the different temperature scales (Kelvin, Celsius, Fahrenheit):

```
> convert_T(1, "C")

       K C    F
1 274.15 1 33.8
```

6.4. salinity and chlorinity

The relationship between Salinity, chlorinity and conductivity is in various functions:

```
> convert_StoCl(S=35)

[1] 19.37394

> convert_RtoS(R=1)

[1] 27.59808

> convert_StoR(S=35)

[1] 1.236537
```

7. finally

This vignette was made with Sweave (Leisch 2002).

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.

FJ M, R F, DG W, TJ M (2008). "The composition of Standard Seawater and the definition of the Reference-Composition Salinity Scale." *Deep-Sea Res. I*, **55**, 50–72.

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.

ME W (2006). "Atomic weights of the elements 2005 (IUPAC Technical Report)." *Pure Appl. Chem.*, **78, No. 11**, 2051Ű2066. URL http://dx.doi.org/10.1351/pac200678112051.

P F, Millard RJ (1983). "Algorithms for computation of fundamental properties of seawater." *Unesco Tech. Pap. in Mar. Sci.*, **44**, 53.

PJ M, Taylor B (2005). "CODATA recommended values of the fundamental physical constants: 2002." *Review of Modern Physics*, **77**, 1 – 107.

Sarmiento J, Gruber N (2006). *Ocean Biogeochemical Dynamics*. Princeton University Press, Princeton.

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.

TJ M, Feistel R, FJ M, Jackett D, Wright D, King B, Marion G, A CCT, P S (2009). *Calculation of the Thermophysical Properties of Seawater, Global Ship-based Repeat Hydrography Manual*. IOCCP Report No. 14, ICPO Publication Series no. 134.

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