marelac : Tools for Aquatic Sciences

Karline Soetaert
NIOZ-Yerseke
The Netherlands

Thomas Petzoldt
Technische Universität Dresden
Germany

Filip Meysman
NIOZ-Yerseke
The Netherlands

Abstract

R package **marelac** (Soetaert, Petzoldt, and Meysman 2010) contains chemical and physical constants and functions, datasets, 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, datasets, e.g. atomic weights, gas constants, the earths bathymetry.
- 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 (Fofonoff and Millard 1983) or as from the more recent derivation based on a Gibbs function (Feistel 2008; McDougall, Feistel, Millero, Jackett, Wright, King, Marion, Chen, and Spitzer 2009a).

Package **marelac** does *not* contain chemical functions dealing with the aquatic carbonate system (acidification, pH). These function can be found in two other R packages, i.e. **seacarb** (Lavigne and Gattuso 2010) and **AquaEnv** (Hofmann, Soetaert, Middelburg, and Meysman 2010).

2. Constants and datasets

2.1. Physical constants

Dataset `Constants` contains commonly used physical and chemical constants, as in Mohr 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     8.31447215    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          Faraday constant
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 oceans 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    m3                    total volume of the oceans
3    VolSurf 1.81e+16    m3        volume of the surface ocean (0-50m)
4    VolDeep 9.44e+17    m3          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
```

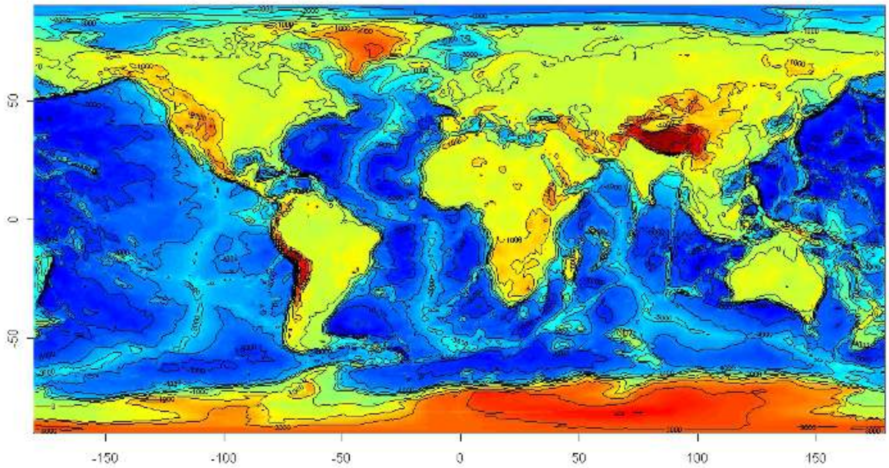

Figure 1: Image plot of ocean bathymetry - see text for R-code

```
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. World bathymetric data

Data set `Bathymetry` from the **marelac** package can be used to generate the bathymetry (and hypsometry) of the world oceans (and land) (Fig.1):

```
require(marelac)
image(Bathymetry$x, Bathymetry$y, Bathymetry$z, col = femmecol(100),
      asp = TRUE, xlab = "", ylab = "")
contour(Bathymetry$x, Bathymetry$y, Bathymetry$z, add = TRUE)
```

2.4. Surface of 1 dg by 1 dg grid cells of the earth

Function `earth_surf` estimates the surface (m^2) of the bathymetric grid cells of 1dg by 1dg, based on their latitude.

As an example, we use it to estimate the surface of the earth; the true surface is 510072000 km^2:

```
> SURF <- outer(X = Bathymetry$x,
+               Y = Bathymetry$y,
+               FUN <- function(X, Y) earth_surf(Y, X))
> sum(SURF)
```

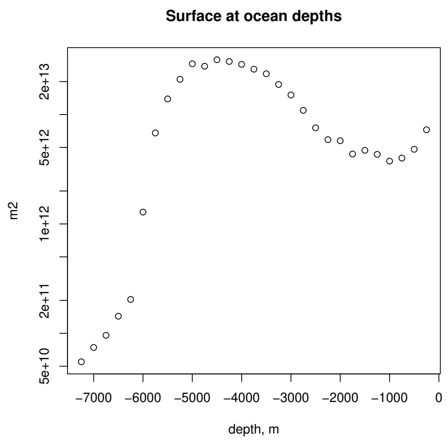

Figure 2: Earth surface at certain ocean depths - see text for R-code

```
[1] 5.100718e+14
```

Similarly, we can estimate the surface and volume of the oceans; it should be $3.58e^{14}$ and $1.34e^{+18}$ respectively.

```
>  sum(SURF*(Bathymetry$z < 0))

[1] 3.618831e+14

> - sum(SURF*Bathymetry$z*(Bathymetry$z < 0))

[1] 1.336255e+18
```

Combined with the dataset `Bathymetry`, function `earth_surf` allows to estimate the total earth surface at certain water depths (Fig.2):

```
> SurfDepth <- vector()
> dseq <- seq(-7500, -250, by = 250)
> for (i in 2:length(dseq)) {
+   ii <- which (Bathymetry$z > dseq[i-1] & Bathymetry$z <= dseq[i])
+   SurfDepth[i-1]<-sum(SURF[ii])
+ }

> plot(dseq[-1], SurfDepth, xlab="depth, m", log = "y",
+      ylab = "m2", main = "Surface at ocean depths")
```

2.5. AtomicWeight

Dataset `AtomicWeight` holds the atomic weight of most chemical elements according to the IUPAC table (Wieser 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 (see also `molweight`).

```
> 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
```

2.6. 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()
```

```
         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 $\omega = 7.292e^{-5}\text{radians}\,\text{sec}^{-1}$.

The following R-script plots the coriolis factor as a function of latitude (Fig.3):

```
> 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 (m^2s^{-1}), for several species at given salinity (S) temperature (t) and pressure (P) is estimated. The implementation is based on Chapter 4 in (Boudreau 1997).

```
> diffcoeff(S = 15, t = 15)*24*3600*1e4  # cm2/day
```

```
       H2O       O2      CO2       H2      CH4       DMS       He       Ne
1 1.478897 1.570625 1.241991 3.429952 1.198804 0.8770747 5.186823 2.749411
        Ar       Kr        Xe        Rn       N2      H2S      NH3
1 1.554712 1.166917 0.9126865 0.8079991 1.190863 1.180685 1.467438
        NO      N2O    CO     SO2       OH         F       Cl       Br
1 1.500764 1.164872 1.195 1.03556 3.543847 0.9577852 1.354384 1.391657
         I      HCO3       CO3     H2PO4     HPO4       PO4       HS
1 1.364436 0.7693272 0.6126977 0.6168857 0.495435 0.3991121 1.214088
       HSO3       SO3      HSO4      SO4       IO3      NO2      NO3
1 0.8836584 0.7379176 0.8874275 0.700226 0.7069267 1.278582 1.283189
         H        Li        Na         K       Cs       Ag      NH4
1 6.510175 0.6738419 0.8807268 0.8807268 1.385375 1.106039 1.314599
         Ca        Mg        Fe        Mn        Ba        Be        Cd
1 0.5264259 0.4682133 0.4657005 0.4610938 0.5611859 0.3911549 0.4682133
```

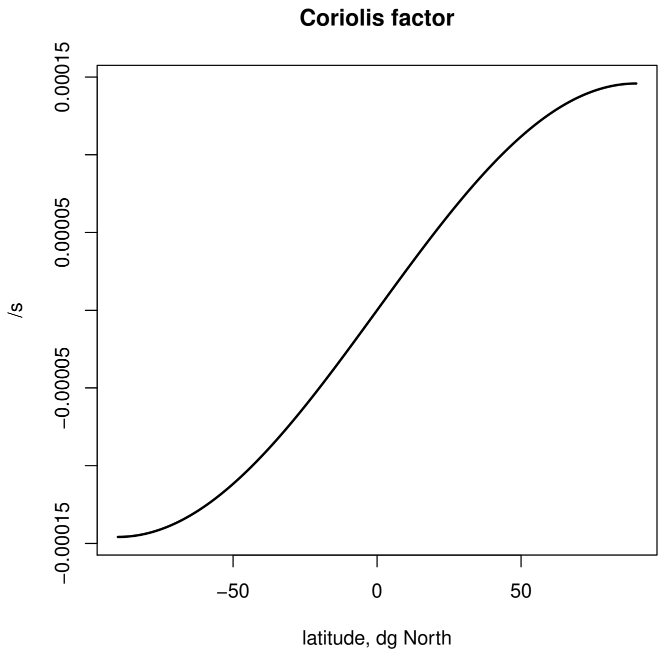

Figure 3: The Coriolis function

```
         Co        Cu        Hg        Ni        Sr      Pb        Ra
1 0.4682133 0.4824524 0.5653738 0.4447608 0.5214003 0.62233 0.5775189
         Zn        Al        Ce        La        Pu    H3PO4      BOH3
1 0.4669569 0.4497863 0.4116759 0.4037188 0.3777535 0.555835 0.7602404
       BOH4    H4SiO4
1 0.6652104 0.6882134
```

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

```
[1] 1.539783e-09
```

Values of the diffusion coefficients for a temperature range of 0 to 30 and for the 13 first species is in (Fig.4):

```
> difftemp <- diffcoeff(t = 0:30)[ ,1:13]
```

```
> matplot(0:30, difftemp, xlab = "temperature", ylab = " m2/sec",
+         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

Function `viscosity` calculates the shear viscosity of water, in centipoise ($gm^{-1}sec^{-1}$). The formula is valid for $0 < t < 30$ and $0 < S < 36$ (Fig.5).

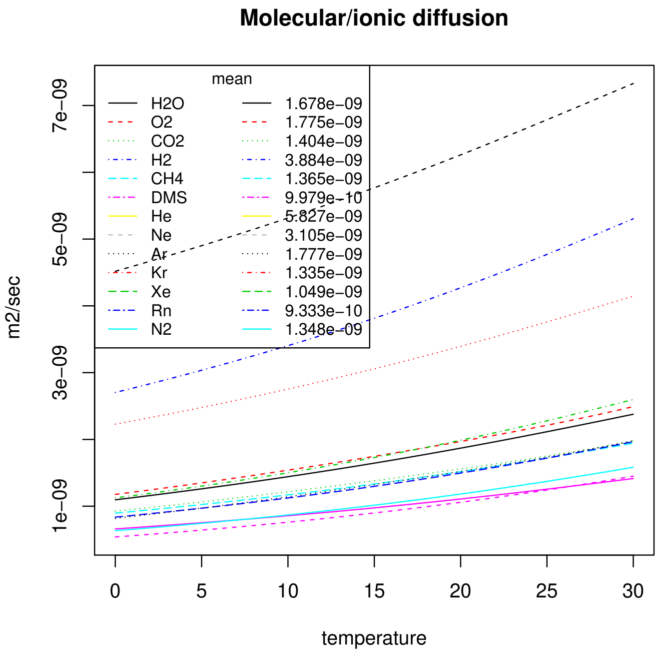

Figure 4: Molecular diffusion coefficients as a function of temperature

```
> plot(0:30, viscosity(S = 35, t = 0:30, P = 1), xlab = "temperature",
+       ylab = "g/m/s", 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 in mgL^{-1}. Method APHA (Greenberg 1992) is the standard formula in Limnology, the method after Weiss (1970) the traditional formula used in marine sciences.

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

```
[1] 7.374404
```

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

Conversion to mmol m^{-3} can be done as follows:

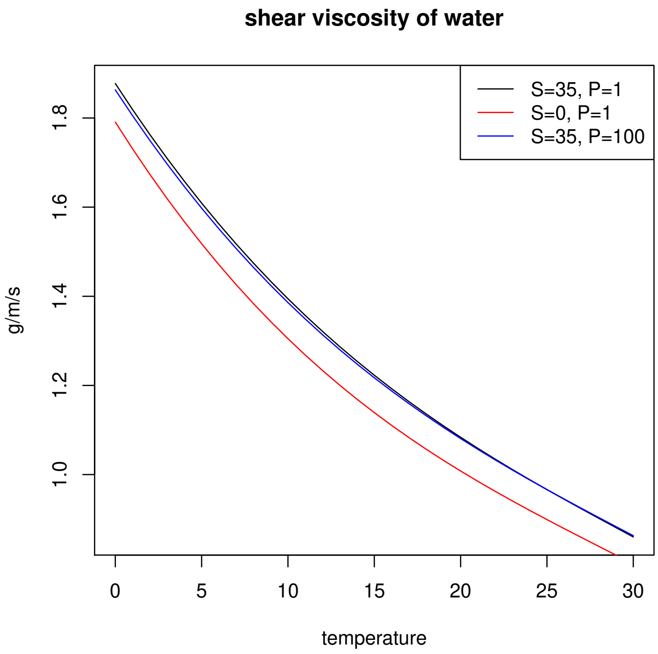

Figure 5: Shear viscosity of water as a function of temperature

```
> gas_O2sat(S=35, t=20)*1000/molweight("O2")

      O2
230.4588
```

The effect of salinity on saturated concentration is in (Fig.6).

```
> plot(t, gas_O2sat(t = t), type = "l", ylim = c(0, 15), lwd = 2,
+      main = "Oxygen saturation", ylab = "mg/l", xlab = "temperature")
> 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)
> legend("topright", c("S=35", "S=0"), col = c("red","green"),
+        lty = c(1, 2), lwd = 2)
```

4.2. Solubilities and saturated concentrations

More solubilities and saturated concentrations (in mmolm^{-3}) are in functions `gas_solubility` and `gas_satconc`.

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

      O2
210.9798
```

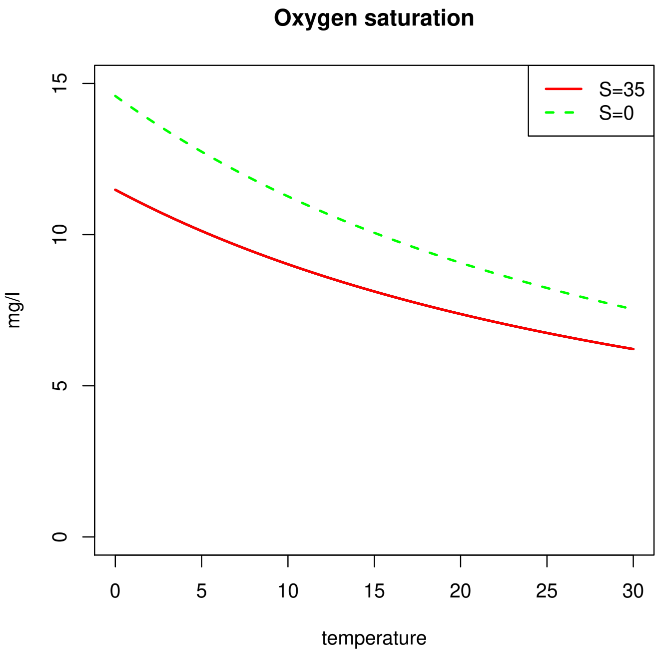

Figure 6: Oxygen saturated concentration as a function of temperature, and for different salinities

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

We plot the saturated concentrations for a selection of species as a function of temperature and salinity (Fig.7):

```
> #
> mf  <-par(mfrow = c(2, 2))
> #
> gs  <-gas_solubility(t = Temp)
> species   <- c("CCl4", "CO2", "N2O", "Rn", "CCl2F2")
> matplot(Temp, gs[, species], type = "l", lty = 1, lwd = 2, xlab = "temperature",
+       ylab = "mmol/m3", main = "solubility (S=35)")
> legend("topright", col = 1:5, lwd = 2, legend = species)
> #
> species2 <- c("Kr", "CH4", "Ar", "O2", "N2", "Ne")
> matplot(Temp, gs[, species2], type = "l", lty = 1, lwd = 2, xlab = "temperature",
+       ylab = "mmol/m3", main = "solubility (S=35)")
> legend("topright", col = 1:6, lwd = 2, legend = species2)
> #
>
> species <- c("N2", "CO2", "O2", "CH4", "N2O")
> gsat  <-gas_satconc(t = Temp, species = species)
```

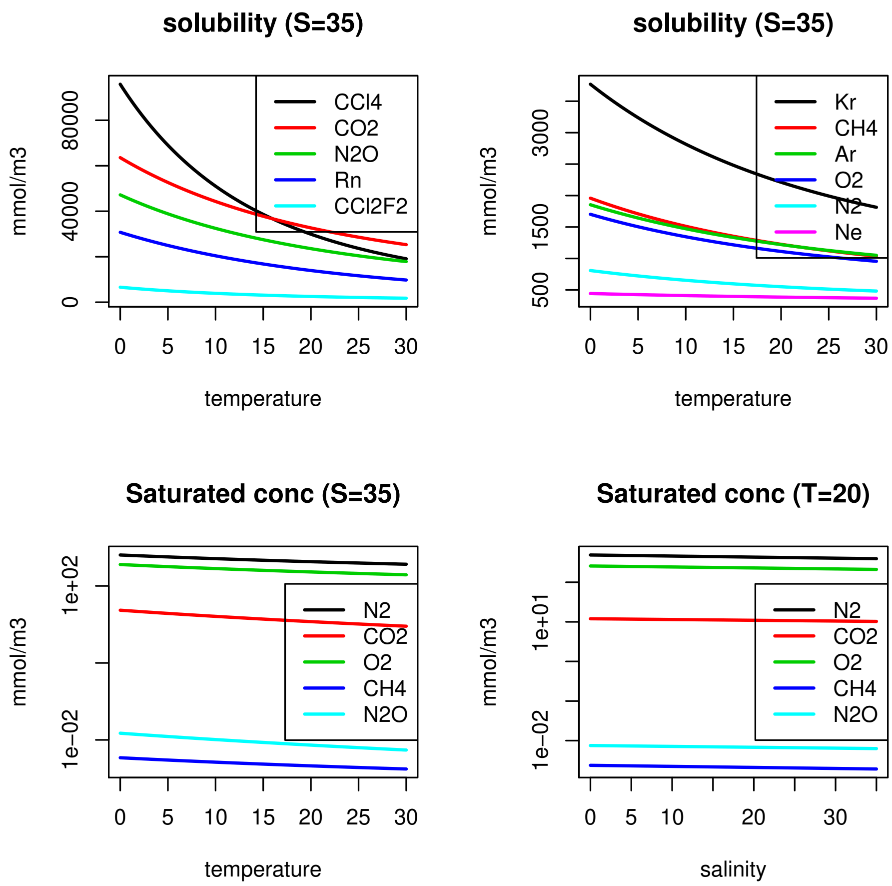

Figure 7: Saturated concentrations and solubility as a function of temperature and salinity, and for different species

```
> 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 = species)
> #
> gsat  <-gas_satconc(S = Sal, species = species)
> 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 = species)
> #
> par("mfrow" = mf)
```

4.3. Partial pressure of water vapor

Function `vapor` estimates the partial pessure of water vapor, divided by the atmospheric pressure (Fig.8).

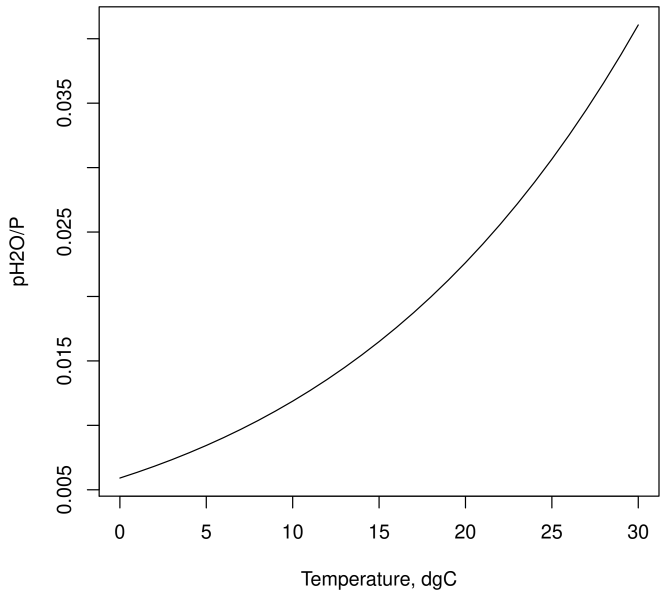

Figure 8: Partial pressure of water in saturated air as a function of temperature

```
> 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 sea level (u_{10})) (Fig.9).

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

         CO2
[1,] 665.988

> useq <- 0:15

> plot(useq, gas_transfer(u10 = useq, species = "O2"), type = "l",
+      lwd = 2, xlab = "u10,m/s", ylab = "m/s",
+      main = "O2 gas transfer velocity", ylim = c(0, 3e-4))
> lines(useq, gas_transfer(u10 = useq, species = "O2", method = "Nightingale"),
+       lwd = 2, lty = 2)
> lines(useq, gas_transfer(u10 = useq, species = "O2", method = "Wanninkhof1"),
+       lwd = 2, lty = 3)
```

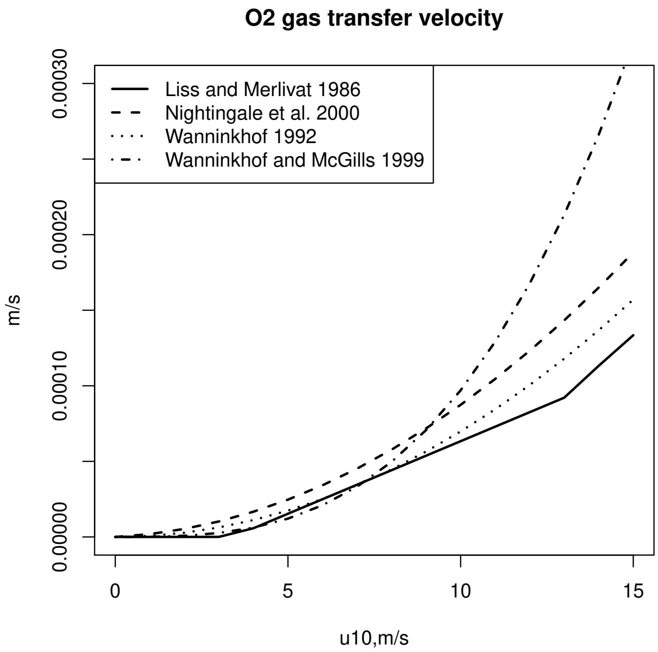

Figure 9: Oxygen gas transfer velocity for seawater, as a function of wind speed

```
> lines(useq, gas_transfer(u10 = useq, species = "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
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. Two salinity scales

Millero, Feistel, Wright, and McDougall (2008) and McDougall, Jackett, and Millero (2009b) provide a function to derive absolute salinity (expressed in g kg^{-1}) from measures of practical salinity. Absolute salinity is to be used as the concentration variable entering the thermodynamic functions of seawater (see next section).

The conversion between salinity scales is done with functions:

- `convert_AStoPS` from absolute to practical salinity and
- `convert_PStoAS` from practical to absolute salinity

For example:

```
> convert_AStoPS(S = 35)

[1] 34.83573

> convert_PStoAS(S = 35)

[1] 35.16504
```

These functions have as extra arguments the gauge pressure (`p`), latitude (`lat`), longitude (`lon`), and -optional- the dissolved Si concentration (`DSi`) and the ocean (`Ocean`).

When one of these arguments are provided, they also correct for inconsistencies due to local composition anomalies.

When `DSi` is not given, the correction makes use of a conversion table that estimates the salinity variations as a function of present-day local seawater composition. The conversion in R uses the FORTRAN code developed by D. Jackett (`http://www.marine.csiro.au/~jackett/TEOS-10/`).

The correction factors are in a data set called `sw_sfac`, a list with the properties used in the conversion functions.

Below we first convert from practical to absolute salinity, for different longitudes, and then plot the correction factors as a function of latitude and longitude and at the seawater surface, i.e. for `p=0` (Fig.10).[1].

```
> convert_PStoAS(S = 35, lat = -10, lon = 0)

[1] 35.16525

> convert_PStoAS(S = 35, lat = 0, lon = 0)

[1] 35.16558
```

[1] Before plotting, the negative numbers in the salinity anomaly table `sw_sfac` are converted to `NA` (not available). In the data set, numbers not available are denoted with `-99`.

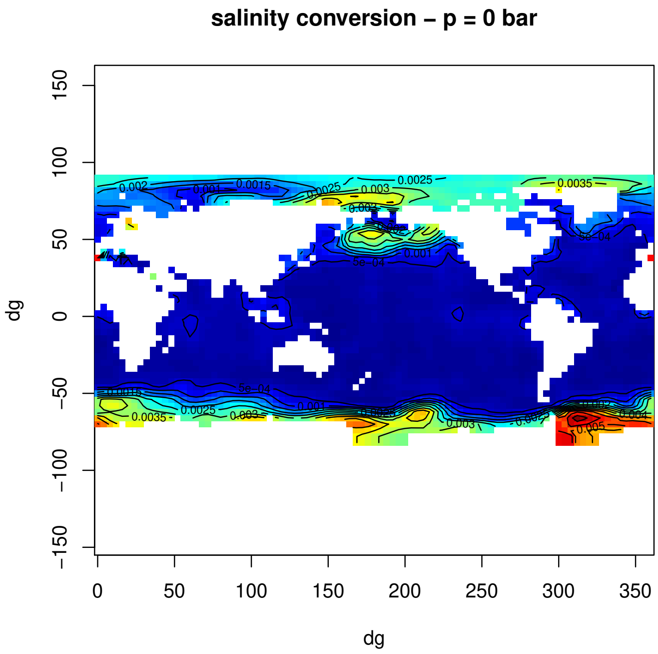

Figure 10: Salinity anomaly to convert from practical to absolute salinity and vice versa

```
> convert_PStoAS(S = 35, lat = 10, lon = 0)

[1] 35.16504

> convert_PStoAS(S = 35, lat = -10, lon = 0)

[1] 35.16525

> convert_PStoAS(S = 35, lat = -10, DSi = 1:10, Ocean = "Pacific")

 [1] 35.16513 35.16523 35.16532 35.16541 35.16550 35.16560 35.16569
 [8] 35.16578 35.16588 35.16597

> dsal <- t(sw_sfac$del_sa[1, , ])
> dsal [dsal < -90] <- NA

> image(sw_sfac$longs, sw_sfac$lats, dsal, col = femmecol(100),
+       asp = TRUE, xlab = "dg", ylab = "dg",
+       main = "salinity conversion - p = 0 bar")
> contour(sw_sfac$longs, sw_sfac$lats, dsal, asp = TRUE, add = TRUE)
```

Finally, the correction factors are plotted versus depth, at four latitudinal cross-sections (Fig.11):

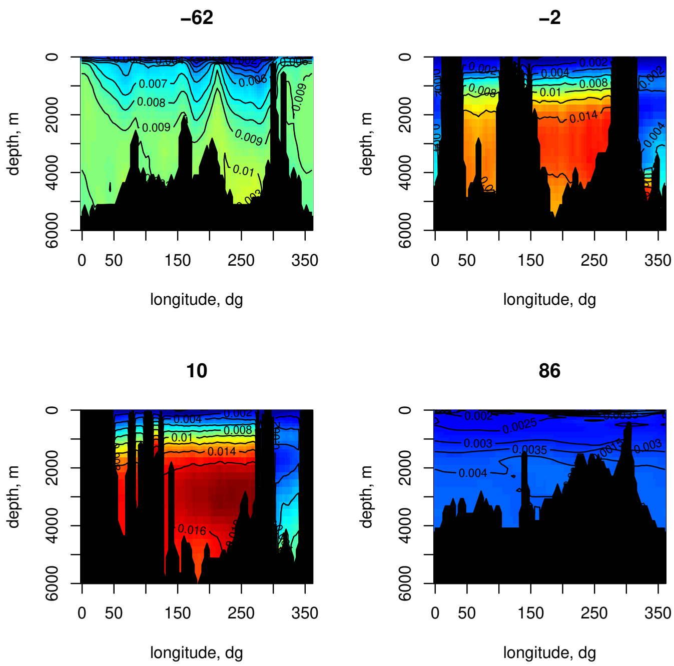

Figure 11: Salinity anomaly to convert from practical to absolute salinity and vice versa for several latitudinal cross-sections (negative = S hemisphere)

```
> ii <- c(6, 21, 24, 43)
> par(mfrow = c(2, 2))
> for ( i in ii)
+ {
+ dsal <- t(sw_sfac$del_sa[ ,i, ])
+ dsal [dsal < -90] <- 0
+ image(sw_sfac$longs, sw_sfac$p, dsal, col = c("black", femmecol(100)),
+       xlab = "longitude, dg", ylab = "depth, m", zlim = c(0, 0.018),
+       main = sw_sfac$lat[i], ylim = c(6000, 0))
+ contour(sw_sfac$longs, sw_sfac$p, dsal, asp = TRUE, add = TRUE)
+ }
```

5.3. Thermophysical seawater properties

Package **marelac** also implements several thermodynamic properties of seawater. Either one can choose the formulation based on the UNESCO polynomial (Fofonoff 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 from Feistel (2008) use the absolute salinity scale (Millero *et al.*

2008), while the UNESCO polynomial uses practical salinity.

```
> 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, method = "UNESCO")

 [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 precision of the calculations can be assessed by comparing them to some test values:

```
> t  <-  25.5
> p  <- 1023/10  # pressure in bar
> S  <- 35.7
> sw_alpha(S, t, p)              -0.0003098378393192645

[1] 1.167598e-13

> sw_beta(S, t, p)               -0.0007257297978386655

[1] 2.555374e-12

> sw_cp(S,t, p)                  -3974.42541259729

[1] -5.945121e-07

> sw_tpot(S, t, p)               -25.2720983155409

[1] 5.203708e-05

> sw_dens(S, t, p)               -1027.95249315662

[1] 9.467044e-08

> sw_enthalpy(S, t, p)           -110776.712408975

[1] -2.050104e-05

> sw_entropy(S, t, p)            -352.81879771528

[1] -9.916204e-08
```

```
> sw_kappa(S, t, p)             -4.033862685464779e-6

[1] -1.068645e-15

> sw_kappa_t(S, t, p)           -4.104037946151349e-6

[1] -1.011721e-15

> sw_svel(S, t, p)              -1552.93372863425

[1] 1.341919e-07
```

Below we plot all implemented thermophysical properties as a function of salinity and temperature (Fig.12, 13). We first define a function that makes the plots:

```
> 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))
```

The difference between the two formulations, based on the UNESCO polynomial or the Gibss function is also instructive (Fig.14):

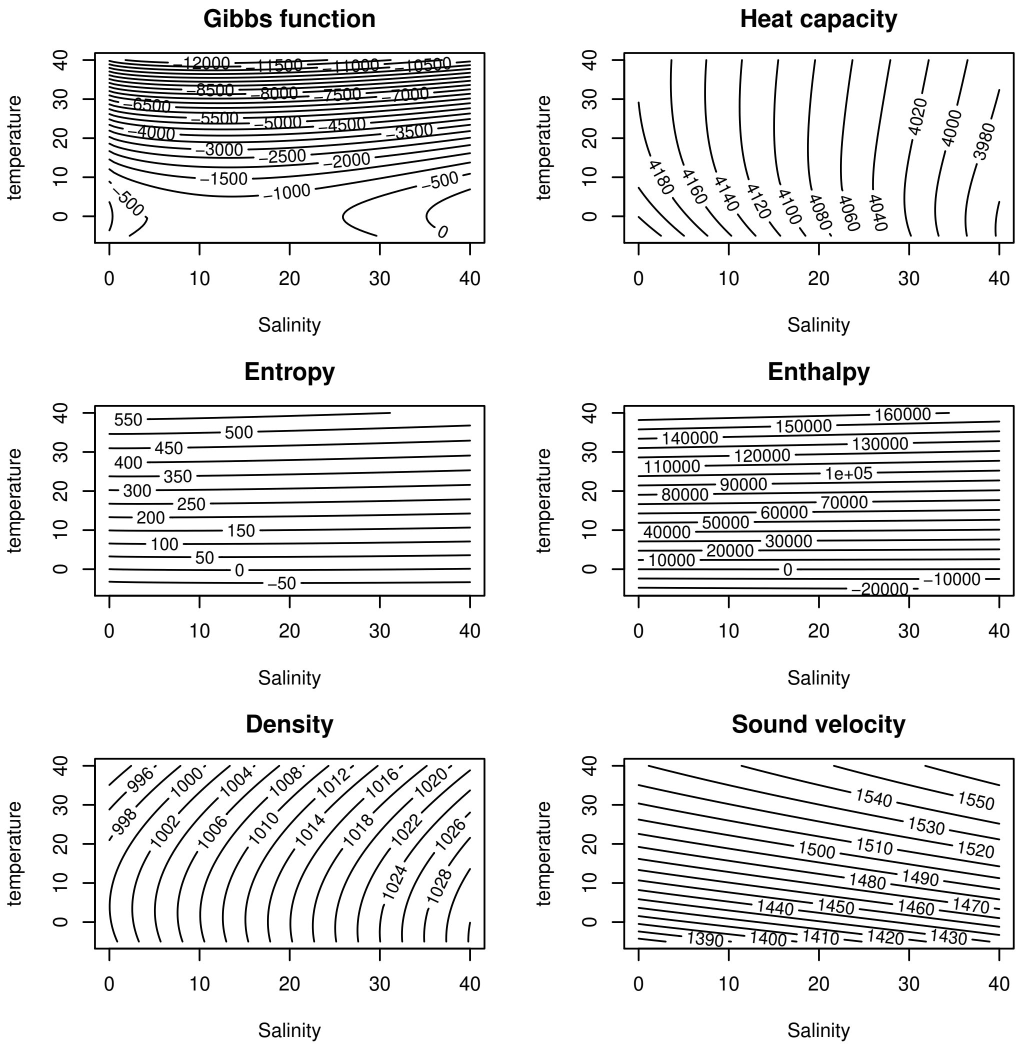

Figure 12: Seawater properties as a function of salinity and temperature - see text for R-code

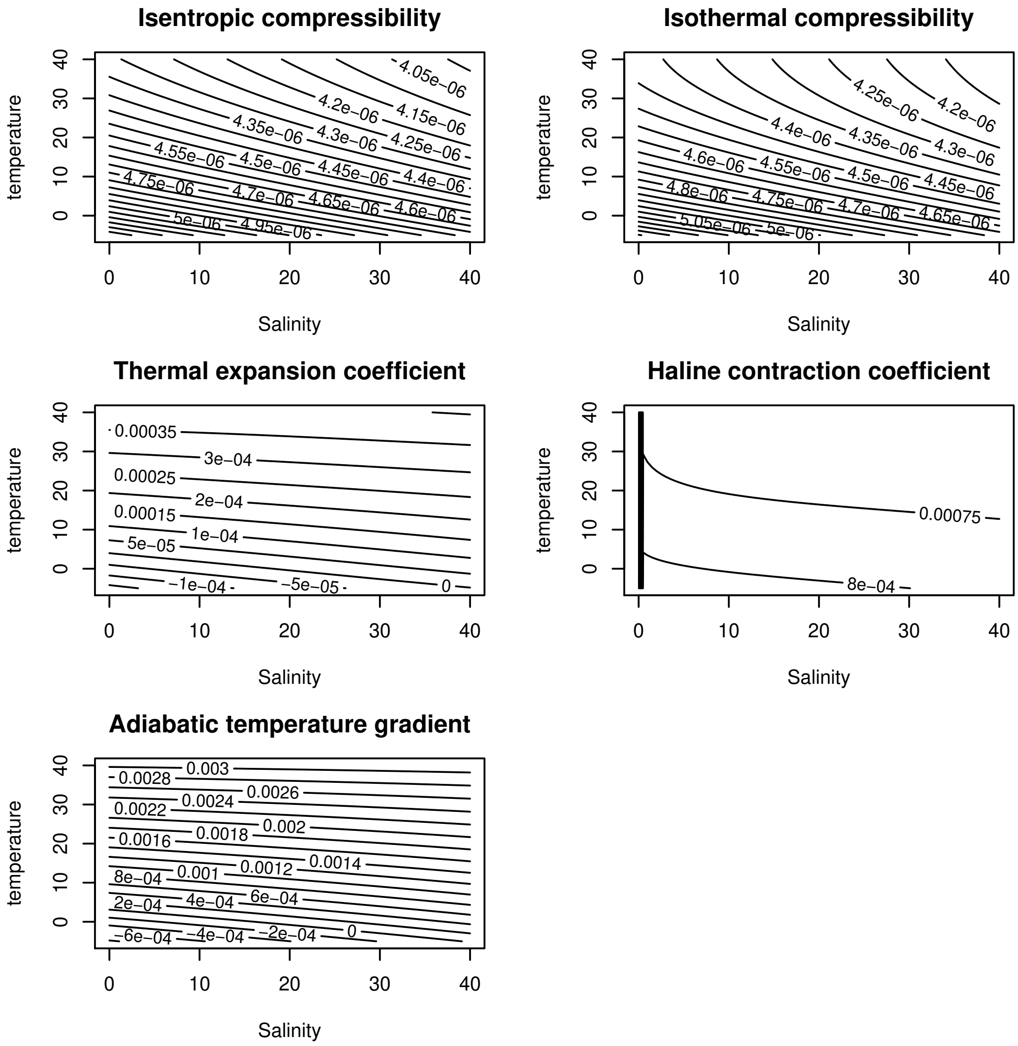

Figure 13: Seawater properties as a function of salinity and temperature - continued - see text for R-code

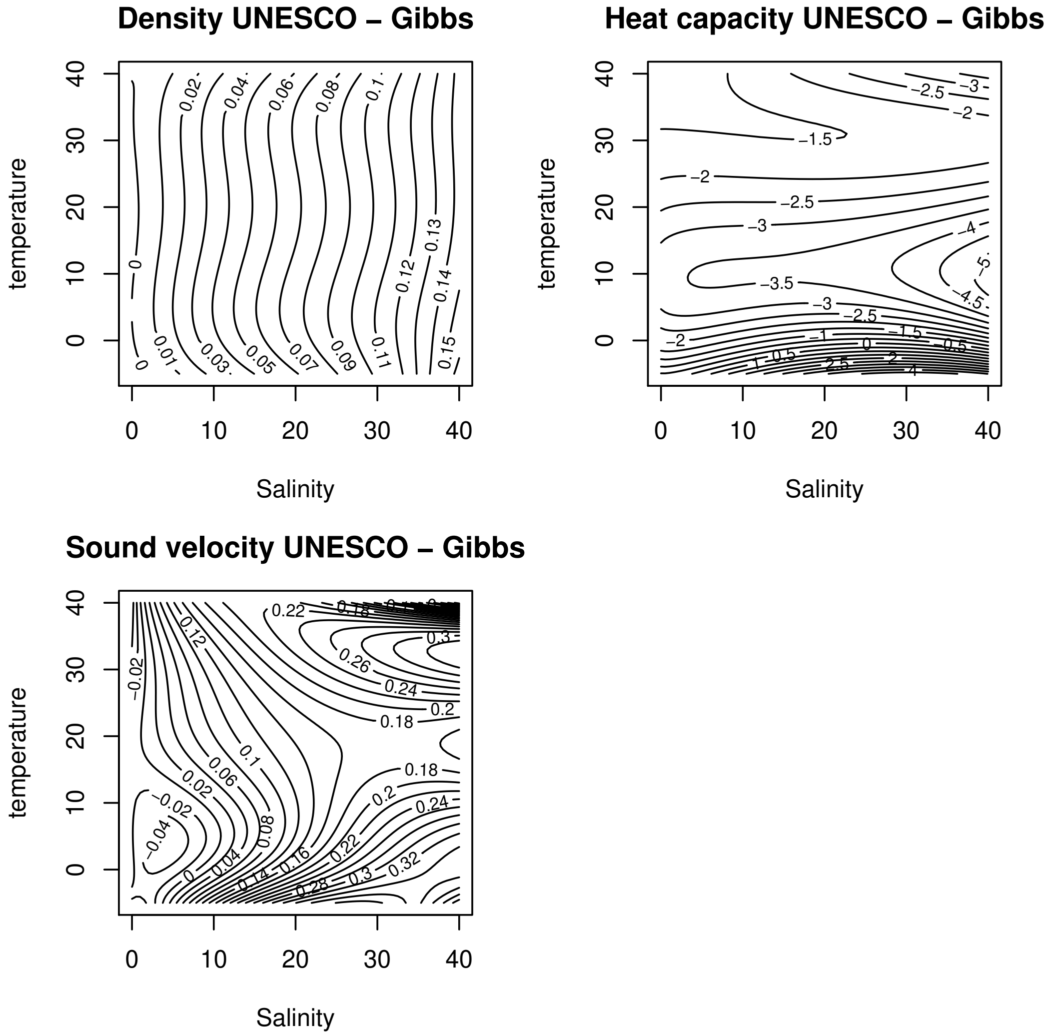

Figure 14: Difference between two methods of calculating some seawater properties as a function of salinity and temperature - see text for R-code

```
> par (mfrow = c(2, 2))
> par(mar = c(4, 4, 3, 2))
> plotST(function(S, t) sw_dens(S, t, method = "UNESCO") - sw_dens(S, t),
+   "Density UNESCO - Gibbs")
> plotST(function(S, t) sw_cp(S, t, method = "UNESCO") - sw_cp(S, t),
+   "Heat capacity UNESCO - Gibbs")
> plotST(function(S, t) sw_svel(S, t, method = "UNESCO") - sw_svel(S, t),
+   "Sound velocity UNESCO - Gibbs")
> 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. The function is based on a lexical parser and the IUPAC table of atomic weights, so it should be applicable to arbitrary chemical formulae:

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

       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 (Fig.15):

```
> #species <- colnames(gs)  ## thpe: does not work any more, because 1D return value is ve
> species = c("He", "Ne", "N2", "O2",
+   "Ar", "Kr", "Rn", "CH4", "CO2", "N2O", "CCl2F2", "CCl3F", "SF6", "CCl4")
> gs <- gas_solubility(species = species)
> mw <- molweight(species)

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

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

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

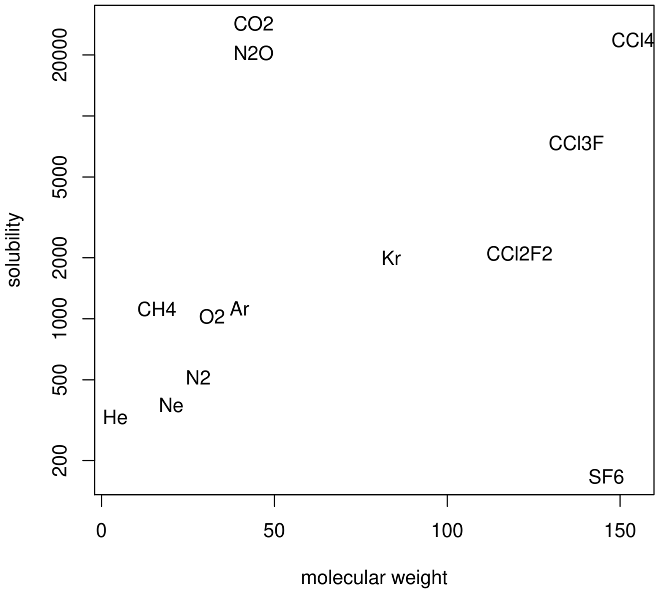

Figure 15: Gas solubility as a function of molecular weight see text for R-code

```
   ideal
24.46536
```

```
> molvol(species = "ideal", t = 1:10)
```

```
         ideal
 [1,] 22.49599
 [2,] 22.57804
 [3,] 22.66010
 [4,] 22.74216
 [5,] 22.82421
 [6,] 22.90627
 [7,] 22.98833
 [8,] 23.07039
 [9,] 23.15244
[10,] 23.23450
```

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

```
      O2
44.67259
```

```
> 1/molvol(species = "O2", q = 1:6, t = 0)
```

```
               O2
[1,] 0.044672589
[2,] 0.022336294
[3,] 0.014890860
[4,] 0.011168149
[5,] 0.008934518
[6,] 0.007445432
```

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

```
            CO2         O2        N2O
[1,] 0.04468587 0.04450899 0.04469987
[2,] 0.04452145 0.04434659 0.04453529
[3,] 0.04435824 0.04418537 0.04437192
[4,] 0.04419623 0.04402533 0.04420975
[5,] 0.04403541 0.04386644 0.04404877
```

6.2. Average elemental composition of biomass

The average elemental composition of marine plankton (Redfield ratio) is traditionally assumed to be $C_{106}H_{263}O_{110}N_{16}P_1$ (Redfield 1934; Redfield, Ketchum, and Richards 1963; Richards 1965), while Limnologists sometimes assume a ratio of $C_{106}H_{180}O_{45}N_{16}P_1$ (Stumm 1964). Since then, the ratio of C:N:P was widely agreed, but there is still discussion about the average of O and H. Anderson (1995) proposed a new formula $C_{106}H_{175}O_{42}N_{16}P_1$ for marine plankton and similarly Hedges, Baldock, Gélinas, Lee, Peterson, and Wakeham (2002), who used NMR analysis, found an elemental composition with much less hydrogen and oxygen ($C_{106}H_{175-180}O_{35-40}N_{15-20}S_{0.3-0.5}$) than in the original formula.

Function `redfield` can be used to simplify conversions between the main elements of biomass, where the default molar ratio can be displayed by:

```
> redfield(1, "P")
```

```
    C   H   O  N P
1 106 263 110 16 1
```

The second argument of the function allows to rescale this to any of the constitutional elements, e.g. to nitrogen:

```
> redfield(1, "N")
```

```
      C       H     O N      P
1 6.625 16.4375 6.875 1 0.0625
```

In addition, it is also possible to request the output in mass units, e.g. how many mass units of the elements are related to 2 mass units (e.g. mg) of phosphorus:

```
> redfield(2, "P", "mass")

         C        H        O        N P
1 82.20727 17.11695 113.6403 14.47078 2
```

Finally, mass percentages can be obtained by:

```
> x <- redfield(1, "P", "mass")
> x / sum(x)

          C         H         O          N           P
1 0.3583026 0.0746047 0.4953044 0.06307127 0.008717054
```

or by using an alternative alternative elemental composition with:

```
> stumm <- c(C = 106, H = 180, O = 45, N = 16, P = 1)
> x <- redfield(1, "P", "mass", ratio = stumm)
> x / sum(x)

          C          H         O          N          P
1 0.5240061 0.07467398 0.2963318 0.09223971 0.01274841
```

Note however, that all these formulae are intended to approximate the **average** biomass composition and that large differences are natural for specific observations, depending on the involved species and their physiological state.

6.3. 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
```

6.4. Temperature conversions

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

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

       K C    F
1 274.15 1 33.8
```

6.5. 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

Anderson LA (1995). "On the hydrogen and oxygen content of marine plankton." *Deep-Sea Res*, **42**, 1675–1680.

Boudreau B (1997). *Diagenetic Models and their Implementation. Modelling Transport and Reactions in Aquatic Sediments.* Springer, Berlin.

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." *Unesco Tech. Pap. in Mar. Sci.*, **44**, 53.

Greenberg AE (1992). *Standard Methods for the Examination of Water and Wastewater.* American Public Health Association, Inc., 18 edition.

Hedges JI, Baldock JA, Gélinas Y, Lee C, Peterson ML, Wakeham SG (2002). "The biochemical and elemental compositions of marine plankton: A NMR perspective." *Marine Chemistry*, **78**(1), 47–63.

Hofmann AF, Soetaert K, Middelburg JJ, Meysman FJR (2010). "AquaEnv - An Aquatic Acid-Base Modelling Environment in R." *Aquatic Geochemistry*, **DOI 10.1007/s10498-009-9084-1**.

Lavigne H, Gattuso J (2010). ***seacarb**: seawater carbonate chemistry with R.* R package version 2.3.3, URL http://CRAN.R-project.org/package=seacarb.

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.

McDougall T, Feistel R, Millero F, Jackett D, Wright D, King B, Marion G, Chen CT, Spitzer P (2009a). *Calculation of the Thermophysical Properties of Seawater, Global Ship-based Repeat Hydrography Manual.* IOCCP Report No. 14, ICPO Publication Series no. 134.

McDougall T, Jackett D, Millero F (2009b). "An algorithm for estimating Absolute Salinity in the global ocean." *Ocean Science Discussions*, **6**, 215–242. URL http://www.ocean-sci-discuss.net/6/215/2009/.

Millero F, Feistel R, Wright D, McDougall T (2008). "The composition of Standard Seawater and the definition of the Reference-Composition Salinity Scale." *Deep-Sea Res. I*, **55**, 50–72.

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

Redfield AC (1934). *in James Johnstone Memorial Volume (ed. Daniel, R. J.).* Liverpool University Press.

Redfield AC, Ketchum BH, Richards FA (1963). "The Sea." In "The influence of organisms on the composition of seawater," volume 2, pp. 26–77. Interscience, New York.

Richards FA (1965). "Anoxic basins and fjords." In JP Riley, D Skirrow (eds.), "Chemical Oceanography," volume 1, pp. 611–645. Academic Press, New York.

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

Soetaert K, Petzoldt T, Meysman F (2010). ***marelac**: Tools for Aquatic Sciences*. R package version 2.1, URL http://CRAN.R-project.org/package=marelac.

Stumm W (1964). "Discussion (Methods for the removal of phosphorus and nitrogen from sewage plant effluents by G. A. Rohlich)." In WW Eckenfelder (ed.), "Advances in water pollution research. Proc. 1st Int. Conf. London 1962," volume 2, pp. 216–229. Pergamon.

Weiss R (1970). "The solubility of nitrogen, oxygen, and argon in water and seawater." *Deep-Sea Research*, **17**, 721–35.

Wieser M (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.

Affiliation:

Karline Soetaert
Royal Netherlands Institute of Sea Research (NIOZ)
4401 NT Yerseke, Netherlands
E-mail: karline.soetaert@nioz.nl
URL: http://http://www.nioz.nl/

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
Royal Netherlands Institute of Sea Research (NIOZ)
4401 NT Yerseke, Netherlands
E-mail: filip.meysman@nioz.nl