Package ‘marelac’

February 20, 2020

Version 2.1.10
Title Tools for Aquatic Sciences
Author Karline Soetaert [aut, cre],
    Thomas Petzoldt [aut],
    Filip Meysman [cph],
    Lorenz Meire [cph]
Maintainer Karline Soetaert <karline.soetaert@nioz.nl>
Depends R (>= 3.2), shape
Imports stats, seacarb
Description Datasets, constants, conversion factors, and utilities for 'MArine', 'Riverine',
    'Estuarine', 'LAcustrine' and 'Coastal' science.
    The package contains among others: (1) chemical and
    physical constants and datasets, e.g. atomic weights, gas
    constants, the earths bathymetry; (2) conversion factors
    (e.g. gram to mol to liter, barometric units,
    temperature, salinity); (3) physical functions, e.g. to
    estimate concentrations of conservative substances, gas
    transfer and diffusion coefficients, the Coriolis force
    and gravity; (4) thermophysical properties of the
    seawater, as from the UNESCO polynomial or from the more
    recent derivation based on a Gibbs function.
License GPL (>= 2)
LazyData yes
Repository CRAN
Repository/R-Forge/Project marelac
Repository/R-Forge/Revision 205
Repository/R-Forge/DateTimeStamp 2020-02-11 22:10:45
Date/Publication 2020-02-20 18:50:04 UTC
NeedsCompilation yes
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>marelac-package</td>
<td>3</td>
</tr>
<tr>
<td>air_density</td>
<td>5</td>
</tr>
<tr>
<td>air_spectrnum</td>
<td>6</td>
</tr>
<tr>
<td>atmComp</td>
<td>7</td>
</tr>
<tr>
<td>AtomicWeight</td>
<td>8</td>
</tr>
<tr>
<td>Bathymetry</td>
<td>9</td>
</tr>
<tr>
<td>Constants</td>
<td>10</td>
</tr>
<tr>
<td>convert_p</td>
<td>11</td>
</tr>
<tr>
<td>convert_RtoS</td>
<td>11</td>
</tr>
<tr>
<td>convert_salinity</td>
<td>13</td>
</tr>
<tr>
<td>convert_StoCl</td>
<td>15</td>
</tr>
<tr>
<td>convert_StoR</td>
<td>16</td>
</tr>
<tr>
<td>convert_T</td>
<td>17</td>
</tr>
<tr>
<td>coriolis</td>
<td>18</td>
</tr>
<tr>
<td>diffcoeff</td>
<td>19</td>
</tr>
<tr>
<td>earth_surf</td>
<td>21</td>
</tr>
<tr>
<td>gas_O2sat</td>
<td>23</td>
</tr>
<tr>
<td>gas_satconc</td>
<td>25</td>
</tr>
<tr>
<td>gas_schmidt</td>
<td>27</td>
</tr>
<tr>
<td>gas_solubility</td>
<td>28</td>
</tr>
<tr>
<td>gas_transfer</td>
<td>31</td>
</tr>
<tr>
<td>gravity</td>
<td>33</td>
</tr>
<tr>
<td>molvol</td>
<td>34</td>
</tr>
<tr>
<td>molweight</td>
<td>36</td>
</tr>
<tr>
<td>Oceans</td>
<td>37</td>
</tr>
<tr>
<td>redfield</td>
<td>38</td>
</tr>
<tr>
<td>ssd2rad</td>
<td>39</td>
</tr>
<tr>
<td>sw_adtgrad</td>
<td>40</td>
</tr>
<tr>
<td>sw_alpha</td>
<td>41</td>
</tr>
<tr>
<td>sw_beta</td>
<td>42</td>
</tr>
<tr>
<td>sw_comp</td>
<td>43</td>
</tr>
<tr>
<td>sw_conserv</td>
<td>44</td>
</tr>
<tr>
<td>sw_cp</td>
<td>45</td>
</tr>
<tr>
<td>sw_dens</td>
<td>47</td>
</tr>
<tr>
<td>sw_depth</td>
<td>49</td>
</tr>
<tr>
<td>sw_enthalpy</td>
<td>50</td>
</tr>
<tr>
<td>sw_entropy</td>
<td>51</td>
</tr>
<tr>
<td>sw_gibbs</td>
<td>52</td>
</tr>
<tr>
<td>sw_kappa</td>
<td>54</td>
</tr>
<tr>
<td>sw_kappa_t</td>
<td>55</td>
</tr>
<tr>
<td>sw_sfac</td>
<td>56</td>
</tr>
<tr>
<td>sw_svel</td>
<td>57</td>
</tr>
<tr>
<td>sw_tfreeze</td>
<td>59</td>
</tr>
<tr>
<td>sw_tpot</td>
<td>60</td>
</tr>
<tr>
<td>vapor</td>
<td>61</td>
</tr>
<tr>
<td>vapor.hPa</td>
<td>62</td>
</tr>
</tbody>
</table>
Description

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. from salinity to chlorinity, from mol to gram, etc.,
- utility functions, e.g. to estimate concentrations of conservative substances as a function of salinity, ...

About the symbols used.

Here we adopt the symbolism as in McDougall et al., 2009:

- S for practical (-) or absolute salinity, (g/kg)
- P for absolute (total) pressure (bar)
- p for sea pressure (also called gauge or applied pressure (bar), the pressure relative to P0, one standard atmosphere (=1.01325 bar)
- t for temperature in °C
- T for absolute temperature, in °K; T = t + 273.15

Many of the functions are from the UNESCO 1983 paper, or from Feistel, 2008. Note that in these papers, pressure is expressed in dbar.

Author(s)

Karline Soetaert (Maintainer)

Thomas Petzoldt

with contributions from Lorenz Meire and Filip Meysman
References

For solubilities, atmospheric composition, gas exchange coefficients:

For diffusion coefficients, viscosity

For many other fundamental properties of seawater, either the UNESCO report (1983):
http://unesdoc.unesco.org/images/0005/000598/059832EB.pdf

or the more recent report and papers:


See Also

for seawater properties:
sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot

for atmospheric gasses:
gas_satconc, gas_O2sat, gas_schmidt, gas_solubility, gas_transfer, atmComp, vapor, air_spechum, air_density

conversions:
convert_AStoPS, convert_PStoAS, convert_RtoS, convert_StoCl, convert_StoR, convert_p, convert_T

datasets:
AtomicWeight, Bathymetry, Constants, Oceans

physical properties:
earth_surf, coriolis, viscosity, diffcoeff, ssd2rad, vertmean, gravity

molecular properties:
AtomicWeight, molvol, molweight, redfield
air_density

Examples

```r
## Not run:
## show examples (see respective help pages for details)
example(air_density)
exmple(molvol)

## open the directory with documents
browseURL(paste(system.file(package="marelac"), "/doc", sep=""))

## End(Not run)
```

---

<table>
<thead>
<tr>
<th>air_density</th>
<th>Air Density</th>
</tr>
</thead>
</table>

Description

The density of the air, in kg/m3

Usage

```r
air_density(t = 25, P = 1.013253)
```

Arguments

- **t**: Temperature, °C.
- **P**: True pressure, bar

Value

The air density, in kg/m3

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>, Lorenz Meire <lorenz.meire@nioz.nl>

References


See Also

vapor, air_spechum, gas_02sat, gas_satconc, gas_schmidt, gas_solubility, gas_transfer, atmComp
Examples

air_density(t = 25) # 1.183894
plot(0:30, air_density(t = 0:30), xlab = "Temperature, dgC", ylab = "kg/m3")
plot(x= seq(0.8,1.2, 0.01), y = air_density(P = seq(0.8,1.2, 0.01)),
xlab = "pressure, bar", ylab = "kg/m3")

---

air_spechum  Air specific humidity

Description

The specific humidity of air (mass mixing ratio in wet air), in kg/kg

Usage

air_spechum(t = 25, rh = 50, P = 1.013253)

Arguments

- **t** Temperature, °C.
- **rh** Relative humidity, %
- **P** True pressure, bar

Value

The specific humidity, in kg/kg.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>, Lorenz Meire <lorenz.meire@nioz.nl>

References

http://www.cactus2000.de/uk/unit/masshum.shtml

See Also

vapor, air_density, gas_02sat, gas_satconc, gas_schmidt, gas_solubility, gas_transfer, atmComp

Examples

air_spechum(t = 25, rh = 50)*1000  # 9.7778
plot(0:30, air_spechum(t = 0:30), xlab = "Temperature, dgC", ylab = "kg/m3")
plot(0:100, air_spechum(rh = 0:100), xlab = "percent humidity", ylab = "kg/kg")
atmComp

Atmospheric Gas Composition

Description

Provides the global average atmospheric composition at present day conditions (year 1998). The mixing ratio is generally defined as the ratio of the mass of an atmospheric constituent to the total mass of dry air. If not otherwise indicated, the term mixing ratio normally refers to water vapor. Here however the mixing ratio is provided for all constituents other than water. The mixing ratio is given as a mole fraction, i.e. the mass of each constituent gas (expressed in moles) divided by the total mass of dry air (also expressed in moles).

Usage

atmComp(species = c("He", "Ne", "N2", "O2", "Ar", "Kr", "CH4", "CO2", "N2O", "H2", "Xe", "CO", "O3"))

Arguments

species character vector selecting the gases whose composition should be provided.

Value

A vector providing the mixing ratio of the selected gases.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>, Filip Meysman <filip.meysman@nioz.nl>

References


They cite Weast and Astle (1982) for all gases except CO2, CH4 and N2O. For the latter three greenhouse gases, the 1998 concentrations are taken from Ramaswamy et al., 2001. Note that the sum of all mixing ratios is slightly larger than one, presumably due to the use of increased greenhouse gases values as compared to Weast and Astle (1982). In fact, the mixing ratio are changing slightly each year due to increases in greenhouse gas concentrations.

See Also

gas_O2sat, gas_satconc, gas_schmidt, gas_solubility, gas_transfer, vapor

Examples

atmComp("O2")
atmComp(c("O2", "CH4", "CO2", "N2O"))
atmComp()
sum(atmComp()) # note this is not =1!
AtomicWeight

The Atomic Weights of Chemical Elements

Description

Atomic weights of chemical elements according to the IUPAC table.

Usage

AtomicWeight
atomicweight

Format

The capitalized version AtomicWeight is a data frame containing the IUPAC table of atomic weights. This data frame has following columns: Number, Name, Symbol, Weight, Footnotes.

Note that as in the IUPAC table, the Weight is given as a text rather than as numeric objects. It comprises the standard values and the uncertainties (in parentheses, following the last significant figure to which they are attributed). See IUPAC report for explanation of the Footnotes.

The lower case version atomicweight is a simplified list that only contains the weights (as numeric values) and allows symbolic computations with elements to arrive at molecular weights.

Details

Molecular weights of chemical elements may vary due to different isotope compositions, depending on geology, industrial processes or biological activity. Please consult the IUPAC Technical report about the details. The data set contains NA for elements that have no stable isotopes (except U, Th, Pa).

Author(s)

Thomas Petzoldt

References


See Also

other datasets: Bathymetry, Constants, Oceans, molweight for molecular weight calculations with molecular formulae.
**Examples**

```r
## assess the data in the IUPAC table (with all footnotes)
AtomicWeight[1:20,
AtomicWeight[AtomicWeight$Symbol == "C",]

## use the lower case version for simple calculations:
atomicweight$C
with(atomicweight, C)

## it can also be used to calculate molecular weights
## via symbolic computation
with(atomicweight, H * 2 + O)
```

---

**Bathymetry**

**World Bathymetric Data**

### Description

This dataset contains the elevation of sea (bathymetry) and land (hypsometry) across the globe at 1 degree intervals. Dataset as used by Andersson et al. (2004).

### Usage

Bathymetry

### Format

A list with the bathymetry (depth) and hypsometry (altitude) of the world. It contains:

- **x** the latitude,
- **y** the longitude,
- **z** the height (m).

### Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

### References

Examples

```r
par(mar = c(2,2,2,2))
image(Bathymetry$x, Bathymetry$y, Bathymetry$z, col = femmecol(100),
    asp = TRUE, xlab = "dg", ylab = "dg")
contour(Bathymetry$x, Bathymetry$y, Bathymetry$z, asp = TRUE, add = TRUE)

# remove land
zz <- Bathymetry$z
zz[zz>0] <- 0
image(Bathymetry$x, Bathymetry$y, zz, col = c(femmecol(100), "black"),
    asp = TRUE)
contour(Bathymetry$x, Bathymetry$y, zz, asp = TRUE, add = TRUE)
```

<table>
<thead>
<tr>
<th>Constants</th>
<th>Useful Physical and Chemical Constants</th>
</tr>
</thead>
</table>

Description

Physical and chemical constants useful for aquatic sciences.

Usage

`Constants`

Format

A list specifying the value, the units, and a description for each physical constant.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

`AtomicWeight`, `Bathymetry`, `Oceans`

Examples

```r
data.frame(cbind(acronym = names(Conditions),
    matrix(ncol = 3, byrow = TRUE, data = unlist(Conditions),
    dimnames = list(NULL, c("value", "units", "description"))))
```
**convert_p**

*Conversion Between Different Barometric Units*

**Description**

The function converts between different units of pressure.

**Usage**

```r
convert_p(x, unit = c("Pa", "bar", "at", "atm", "torr"))
```

**Arguments**

- `x` vector of given pressure values,
- `unit` measurement unit of the given value(s).

**Value**

A data frame with converted values.

**References**

http://en.wikipedia.org/wiki/Bar_(unit)

**See Also**

`convert_AStoPS`, `convert_PStoAS`, `convert_RtoS`, `convert_StoCl`, `convert_StoR`, `convert_T`,

**Examples**

```r
convert_p(1, "atm")
convert_p(c(2, 3, 4.5), "bar")
convert_p(1, "atm")$Pa
```

---

**convert_RtoS**

*Conductivity-Salinity Conversion*

**Description**

Estimates the salinity of seawater from conductivity, temperature and pressure. The equation is valid over ranges: temperature from -2 to 35 °C, pressure from 0 to 1000 bar, and salinity from 2 to 42.
Usage

classical.RtoS(R = 1, t = 25, p = max(0, P-1.013253), P = 1.013253)

Arguments

- **R**
  Conductivity ratio, the conductivity at (\(S, t, P\)) divided by the conductivity at \(S = 35, t = 15, p = 0\) [-]

- **t**
  Temperature, °C

- **p**
  Gauge (or applied) pressure, the pressure referenced against the local atmospheric pressure, bar

- **P**
  True pressure, bar

Value

The salinity.

Note

The conductivity ratio for Salinity = 40.0000, t = 40, p = 1000 is 1.888091.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References

http://unesdoc.unesco.org/images/0005/000598/059832EB.pdf

See Also

convert_AStoPS, convert_PStoAS, convert_StoR, convert_StoCl, convert_p, convert_T

Examples

classical.RtoS(R = 1.888091, t = 40, p = 1000)

## Salinity = 40.0000, t = 40, p = 1000, cond = 1.888091
subset.RtoS(R = 1, t = 15, p = 0)

## Check values
classical.RtoS(R = 0.6990725, t = 10, p = 0) # 26.8609
subset.RtoS(R = 0.6990725, t = 10, p = 100) # 26.5072
subset.RtoS(R = 1.1651206, t = 20, p = 100) # 36.3576
**Practical - Absolute Salinity Conversions**

**Description**
Conversion from practical to absolute salinity and vice versa.

**Usage**
```
convert_PStoAS(S = 35, p = max(0, P - 1.013253), P = 1.013253,
lat = NULL, lon = NULL, DSi = NULL,
Ocean = c("Global", "Atlantic", "Pacific", "Indian", "Southern"))
```
```
convert_AStoPS(S = 35, p = max(0, P - 1.013253), P = 1.013253,
lat = NULL, lon = NULL, DSi = NULL,
Ocean = c("Global", "Atlantic", "Pacific", "Indian", "Southern"))
```

**Arguments**
- **S**: Salinity, either practical salinity (convert_PStoAS), dimensionless or absolute salinity (convert_AStoPS, g/kg)
- **p**: gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- **P**: true pressure, bar
- **lat**: latitude (-90 to +90)
- **lon**: longitude (0 to 360)
- **DSi**: the silicate concentration, in micromol/kg
- **Ocean**: the ocean in which the measurement was taken; only used if DSi is not NULL

**Details**
Absolute salinity (g kg⁻¹) is estimated from Practical salinity as:
```
AS= 35.16504 /35 * PS + delt()
```
where delt is the absolute salinity anomaly. There are two ways in which to estimate the salinity anomaly:

1. If DSi is not given a value, then the anomaly is estimated as a function of longitude lon, latitude lat and pressure p, using the lookup table as in sw_sfac.
2. If DSi is given a value, then a regression on it is used, based on the values of Ocean and -except for the "global" ocean- the latitude lat:

"Global" a global estimate is used,
```
delt= 9.824e-5 *DSi.
```
"Southern" the Southern Ocean (lat < -30),
delt= 7.4884e-5 *DSi,
"Pacific" the Pacific Ocean,
delt= 7.4884e-5 *(1 + 0.3622[lat/30 + 1])*DSi,
"Indian" the Indian Ocean,
delt= 7.4884e-5 *(1 + 0.3861[lat/30 + 1])*DSi,
"Atlantic" the Atlantic Ocean,
delt= 7.4884e-5 *(1 + 1.0028[lat/30 + 1])*DSi,

Note that for the Pacific, Indian and Atlantic Ocean regression, the latitude is needed. If lat is NULL then the Global regression will be used.

Value

The absolute salinity (convert_PStoAS) or practical salinity (convert_AStoPS).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

convert_RtoS, convert_StoR, convert_StoCl, convert_p, convert_T,

Examples

# check values: should be 35.7
convert_PStoAS(S = 35.52764437773386, p = 102.3, lon = 201, lat = -21)

# check values: should be 35.52764437773386
convert_AStoPS(S = 35.7, p = 102.3, lon = 201, lat = -21)

#
convert_PStoAS(S = 35)
convert_AStoPS(S = 35)
convert_PStoAS(S = 35, lat = 10, lon = 10, p = 0)

# Based on Si concentration
DSi <- seq(from = 0, to = 200, by = 10)
Global   <- convert_PStoAS(30, DSi = DSi, Ocean = "Global")
convert_StoCl

Salinity-Chlorinity Conversion

Description
Estimates the chlorinity concentration based on salinity

Usage
convert_StoCl(S = 35)

Arguments
S  salinity

Value
The chlorinity concentration, g/kg

Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

References

See Also
correct_AStoPS, convert_PStoAS, convert_RtoS, convert_StoR, convert_p, convert_T,

Examples
convert_StoCl(20)
**convert_StoR**

*Salinity-Conductivity Conversion*

**Description**

Estimates the conductivity ratio from salinity, temperature and pressure.

The equation is valid over ranges of temperature from -2 to 35 °C, pressure of 0 - 1000 bar and salinity 2-42 in the world ocean.

**Usage**

```
convert_StoR(S = 35, t = 25, p = max(0, P-1.013253), P = 1.013253)
```

**Arguments**

- **S** (practical) Salinity, -.
- **t** Temperature, °C,
- **P** True Pressure, bar,
- **p** Gauge (or applied) pressure, the pressure referenced against the local atmospheric pressure, bar.

**Value**

The conductivity ratio (-), this is the conductivity at (S, t, p), divided by the conductivity at S = 35, t = 15, p = 0.

**Note**

Pressure here is true pressure, 1 bar (at sea surface), in contrast to hydrostatic pressure in dbar of original formula.

The conductivity ratio for Salinity = 40, t = 40, p = 1000 is 1.888091.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


**See Also**

`convert_AStoPS, convert_PStoAS, convert_RtoS, convert_StoCl, convert_p, convert_T,`
convert_T

Examples

convert_StoR(S = 40, t = 40, p = 1000)

convert_StoR(S = 35, t = 15, p = 0)

# Check values:
convert_StoR(S = 25, t = 10, p = 0)  # 0.654990
convert_StoR(S = 25, t = 10, p = 100) # 0.662975
convert_StoR(S = 25, t = 10, p = 1000) # 0.712912

convert_StoR(S = 35, t = 10, p = 100) # 0.897778
convert_StoR(S = 40, t = 10, p = 100) # 1.011334

---

### convert_T

**Conversion Between Different Temperature Units**

**Description**

The function converts between different units of temperature.

**Usage**

```r
convert_T(x, unit = c("K", "C", "F"))
```

**Arguments**

- `x`: Vector of given temperature values,
- `unit`: Measurement unit of the given value(s).

**Value**

A data frame with converted values.

**References**


http://www.cstl.nist.gov/div836/836.05/papers/magnum90ITS90guide.pdf

**See Also**

- `convert_AStoPS`
- `convert_PStoAS`
- `convert_RtoS`
- `convert_StoR`
- `convert_StoCl`
Examples

```r
coriolis(0, "K")
coriolis(0, "C")
coriolis(0, "F")

coriolis(273.15, "K")
coriolis(-273.15, "C")
coriolis(c(-459.67, 0, 32), "F")

coriolis(32, "F")$C # 0 degrees C
```

The Coriolis Force as a Function of Latitude

Description

Estimates the coriolis factor, \( f \) (in \( s^{-1} \)), where \( f = 2 \cdot \omega \cdot \sin(\text{lat}) \), where \( \omega = 7.292e^{-5} \) radians/sec, the rotation of the earth.

Usage

```r
coriolis(lat)
```

Arguments

- `lat` latitude in degrees north (-90 to +90).

Value

The coriolis factor (\( s^{-1} \)).

Author(s)

Karline Soetaert < karline.soetaert@nioz.nl >

References


See Also

`viscosity`, `diffcoeff`, `ssd2rad`, `vertmean`, `gravity`
**Examples**

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

---

**diffcoeff**

**Molecular Diffusion Coefficients**

**Description**

Calculates the molecular and ionic diffusion coefficients in \( m^2 s^{-1} \), for several inorganic species in seawater at a given salinity, temperature, and pressure.

Based on Chapter 4 in Boudreau (1997)

**Usage**

```r
diffcoeff(S = 35, t = 25, P = 1.013253,
    species = c("H2O", "O2", "CO2", "H2", "CH4", "DMS",
    "He", "Ne", "Ar", "Kr", "Xe", "Rn",
    "N2", "H2S", "NH3", "NO", "N20", "CO", "SO2",
    "OH", "F", "Cl", "Br", "I",
    "HCO3", "CO3", "H2P04", "HP04", "P04",
    "HS", "HSO3", "SO3", "HSO4", "S04", "IO3", "NO2", "NO3",
    "H", "Li", "Na", "K", "Cs","Ag","NH4",
    "Ca", "Mg", "Fe", "Mn", "Ba", "Be", "Cd", "Co",
    "Cu", "Hg", "Ni", "Sr", "Pb", "Ra", "Zn", "Al", "Ce",
    "La", "Pu", "H3P04", "BOH3", "B0H4", "H4SiO4")
```

**Arguments**

- **S** : Salinity, -.
- **t** : Temperature, °C.
- **P** : True pressure, bar.
- **species** : character vector with the names of the chemical species whose diffusion coefficient should be calculated.

**Details**

To correct for salinity, the Stokes-Einstein relationship is used. This is not quite accurate, but is at least consistent.

- **\( H_3PO_4 \)**: Least (1984) determined D(H3PO4) at 25 deg C and 0 S. Assume that this value can be scaled by the Stokes-Einstein relationship to any other temperature.
- **\( B(OH)_3 \)**: Mackin (1986) determined D(B(OH)3) at 25 deg C and about 29.2 S. Assume that this value can be scaled by the Stokes-Einstein relationship to any other temperature.
- **\( B(OH)_4 \)**: No information on this species. Boudreau and Canfield (1988) assume it is 12.5% smaller than B(OH)3.
$H_4SiO_4$: Wollast and Garrels (1971) found $D(H_4SiO_4)$ at 25 deg C and 36.1 ppt S. Assume that this value can be scaled by the Stokes-Einstein relationship to any other temperature.

Arguments salinity, temperature or pressure can be vectors. In order to avoid confusion, $S$, $t$ and $P$ must have either same length or length 1. More flexible combinations are of course possible with `expand.grid`.

**Value**

A data.frame with the diffusion coefficients $m^2s^{-1}$ of the selected chemical species.

**Author(s)**

Filip Meysman <filip.meysman@nioz.nl>, Karline Soetaert <karline.soetaert@nioz.nl>

**References**

Based on Chapter 4 in Boudreau (1997):


who cites:

for self-diffusion coefficient H2O:


for gases O2 and CO2:

Novel relation by Boudreau (1997) based on new compilation of data

for gases He, Ne, Kr, Xe, Rn, H2, CH4:


for Ar:


for DMS:


for other gases (N2, H2S, NH3, NO, N2O, CO, SO2):


with the correction proposed by


for ions:
for H3PO4, B(OH)3, B(OH)4, H4SiO4: see details

See Also
coriolis, viscosity, ssd2rad, vertmean, gravity

Examples

diffcoeff(S = 15, t = 15)*1e4*3600*24 # cm2/day
diffcoeff(t = 10, species = "O2") # m2/s
difftemp <- diffcoeff(t = 0:30)[,1:13]
matplot(0:30, difftemp, xlab = "temperature", ylab = "m2/s",
main = "Molecular/ionic diffusion", type = "l",
col = 1:13, lty = 1:13)
legend("topleft", ncol = 2, cex = 0.8, title = "mean",
col = 1:13, lty = 1:13,
legend = cbind(names(difftemp),
format(colMeans(difftemp), digits = 4)))

## vector-valued salinity
select <- c("O2", "CO2", "NH3", "NH4", "NO3")
diffsal <- diffcoeff(S = 0:35, species = select)
matplot(0:35, diffsal, xlab = "salinity", ylab = "m2/s",
main = "Molecular/ionic diffusion", type = "l",
col = 1:length(select), lty = 1:length(select))
legend("topleft", ncol = 2, cex = 0.8, title = "mean",
col = 1:length(select), lty = 1:length(select),
legend = cbind(select, format(colMeans(diffsal), digits = 4)))

## vector-valued temperature
difftemp <- diffcoeff(S = 1, t=1:20, species = select)
matplot(1:20, difftemp, xlab = "temperature", ylab = "m2/s",
main = "Molecular/ionic diffusion", type = "l",
col = 1:length(select), lty = 1:length(select))
legend("topleft", ncol = 2, cex = 0.8, title = "mean",
col = 1:length(select), lty = 1:length(select),
legend = cbind(select, format(colMeans(difftemp), digits = 4)))

## combination of S and t
diffsaltemp <- diffcoeff(S = rep(c(1, 35), each = 20),
t = rep(1:20, 2), species = select)
Description

`earth_surf` computes the surface of 1d by 1dg grid cells as a function of latitude. Based on data that give the surface distance per 1 dg change in lat/lon from [http://en.wikipedia.org/wiki/Latitude](http://en.wikipedia.org/wiki/Latitude)

`earth_dist` calculates the distance between two (lat, lon) points

Usage

```r
earth_surf(lat = 0, lon = 0)
earth_dist(alat, alon, blat, blon, method = 1)
```

Arguments

- `lat` latitude (-90 - +90).
- `lon` longitude - not used.
- `alat` first latitude (-90 - +90).
- `alon` first longitude (-180, 180).
- `blat` second latitude (-90 - +90).
- `blon` second longitude (-180, 180).
- `method` an integer indicating the formula to use, either the spherical law of cosines (1) or the haversine formula (2)

Value

Surface of the grid cell, in \(m^2\).

Distance between the points (alat, alon), (blat, blon), m.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

`Bathymetry`, `Oceans`

Examples

```r
earth_surf(seq(0, 90, by = 15))
SURF <- outer(X = Bathymetry$x,
               Y = Bathymetry$y,
               FUN <- function(X, Y) earth_surf(Y, X))
earth_dist(10, 80, 10, 81)
earth_dist(20, 80, 20, 81)
```
gas_O2sat

\begin{verbatim}
SURF <- outer(X = Bathymetry$x, 
    Y = Bathymetry$y, 
    FUN <- function(X, Y) earth_surf(Y, X))

sum(SURF) # is: 510,072,000 km2

# the surface of the Oceans, m2
sum(SURF*(Bathymetry$z < 0)) # is: 3.58e14

# the volume of the Oceans, m3
- sum(SURF*Bathymetry$z*(Bathymetry$z < 0)) # is: 1.34e+18

# the surface area at several depths
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")
\end{verbatim}

gas_O2sat

\textit{Saturation Concentration of Oxygen in Water}

\textbf{Description}

Empirical formulae that can be used to compute saturation concentration of oxygen in water in \textit{mg/L}

\textbf{Usage}

gas_O2sat(S = 35, t = 25, masl = 0, method = c("Weiss", "APHA", "Paul"))

\textbf{Arguments}

\begin{itemize}
    \item \textbf{S} \hspace{1cm} salinity (dimensionless, for method "Weiss" only),
    \item \textbf{t} \hspace{1cm} Temperature in °C,
    \item \textbf{masl} \hspace{1cm} height above sea level (in m, for method "Paul" only),
    \item \textbf{method} \hspace{1cm} formula to be used, see references for correct application.
\end{itemize}
Details

Method APHA is the standard formula in Limnology, method Weiss the standard formula in marine sciences. The method after Paul is a simple formula fitted on available tables. To avoid confusion between the arguments (S, t, masl) it is advisable to use named arguments in general, e.g. 02sat(t = 4).

Value

Vector with oxygen saturation concentration in \( mgL^{-1} \).

References


DIN 38408-23, Ausgabe:1987-11: Deutsche Einheitsverfahren zur Wasser-, Abwasser- und Schlammmuntersuchung; Gasförmige Bestandteile (Gruppe G); Bestimmung des Sauerstoffsättigungindex (G 23).


See Also

gas_satconc for other gas species and explicit consideration of pressure.

Examples

gas_02sat(S = 0, t = 20) # fresh water, Weiss formula
gas_02sat(S = 0, t = 20, method = "APHA") # fresh water, APHA formula

## compare this with
gas_satconc(S = 0, t = 20, species = "O2") * molweight("O2") / 1000

T <- seq(0, 30, 0.1)
plot(T, gas_02sat(S = 0, t = T, method = "APHA"),
 ylab="O2 sat (mg/L)", type = "l", ylim = c(0, 15))
lines(T, gas_02sat(S = 0, t = T, method = "Weiss"),
col = "blue", lwd = 2, lty = "dashed")
gas_satconc

Saturated Concentrations of Gases in Water

Description

Calculates the saturated concentration of several gases in water for a given temperature, salinity and pressure.

Usage

gas_satconc(S = 35, t = 25, P = 1.013253, 
species = c("He", "Ne", "N2", "O2", "Ar", "Kr", "CH4", "CO2", "N2O"), 
atm = atmComp(species))

Arguments

S Salinity (dimensionless),
t Temperature, °C,
P True pressure, bar
species character vector with gasses whose saturated concentration should be estimated.
atm The number of moles of the gas per unit mole of air in the atmosphere, the "mixing ratio". When present, this overrules the species argument. When unspecified, the value from atmComp for the species is taken.

Value

The saturated concentration of the gas in mmol m\(^{-3}\).

Note

Compared to the table in Sarmiento and Gruber, there is a slight deviation for N2O, and He. CO2 is OK for temperature 0 only.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>
References


who cite:


See Also

gas_O2sat, gas_schmidt, gas_solubility, gas_transfer, atmComp, vapor

Examples

gas_satconc(species = "O2")
Temp <- seq(from = 0, to = 30, by = 0.1)
Sal <- seq(from = 0, to = 35, by = 0.1)

mf <- par(mfrow = c(1,2))

species <- c("N2", "CO2", "O2", "CH4", "N2O")
gsat <- gas_satconc(t = Temp, species = 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)
gas_schmidt

par(mfrow = mf)

## generate table 3.2.4 from Sarmiento and Gruber
Temp <- seq (0, 30, by = 5)
## saturated concentrations in mmol/m3, at 1 atm.
A <- data.frame(cbind( t = Temp,
    N2 = gas_satconc(t = Temp, species = "N2"),
    O2 = gas_satconc(t = Temp, species = "O2"),
    CO2 = gas_satconc(t = Temp, species = "CO2"),
    Ar = gas_satconc(t = Temp, species = "Ar")))
format(A, digits = 4)
## table values
## at 0 dg C: 635.6 359.1 23.37 17.44
## at 20 dg C: 425.7 230.5 11.61 11.29
## note the deviations for CO2 (20dg)!

## saturated concentrations in micromol/m3, at 1 atm.
AA <- data.frame(cbind(t = Temp,
    N2O = gas_satconc(t = Temp, species = "N2O")*1000,
    Ne = gas_satconc(t = Temp, species = "Ne")*1000,
    Kr = gas_satconc(t = Temp, species = "Kr")*1000,
    CH4 = gas_satconc(t = Temp, species = "CH4")*1000,
    He = gas_satconc(t = Temp, species = "He")*1000))
format(AA, digits = 4)
## table values
## at 0 dgC: 14.84 8.11 4.33 3.44 1.81
## at 20 dgC: 7.16 6.94 2.50 2.12 1.70
## Note: different for N2O

gas_schmidt

The Schmidt Number for Gases in Seawater

description

The Schmidt number as a function of temperature (0-30dgC) and for a salinity of 35.

$Sc = \frac{v}{D} = \frac{\mu}{(\rho + D)}$

where v is the kinematic viscosity of the water and D is the mass diffusivity, rho is density and mu is the viscosity.

Schmidt numbers are used to estimate the gas transfer velocity.

usage

Arguments

- `t` Temperature in °C,
- `species` character vector with gases whose schmidt number should be estimated.

Value

The Schmid number, a dimensionless quantity.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


who cite:


except for $O_2$:


CFC-11 ($CCl_2F_2$), and CFC-12 ($CCl_3F$):


and $CCl_4$ (Wanninkhof, pers.comm).

See Also

gas_O2sat, gas_satconc, gas_solubility, gas_transfer, atmComp, vapor

Examples

gas_schmidt(species = "CO2", t = 20) # about 660

---

gas_solubility | Solubility Parameters
---

Description

Solubility parameters $SA$, $mmol m^{-3} bar^{-1}$, calculated from the Bunsen solubility coefficients and the volumetric solubility coefficients.
gas_solubility

Usage

gas_solubility(S = 35, t = 25,
    species = c("He", "Ne", "N2", "O2", "Ar", "Kr", "Rn", "CH4",
                "CO2", "N2O", "CCl2F2", "CCl3F", "SF6", "CCl4"))

Arguments

S  salinity, -
t  temperature, °C,
species  The gas

Value

The solubility, mmol/m3/bar.

Note

The molar volume used for the Bensen coefficient conversion is the ideal gas value of 22.4136 l/mol.

These coefficients are to be used with pAmoist, the partial pressure of the gas in moist air.

To convert them for use with partial pressure in dry air, divide by (1-vapor(S,t)).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


who cite:


See Also
gas_O2sat, gas_satconc, gas_schmidt, gas_transfer, atmComp, vapor

Examples
gas_solubility(t = 1:20, S = 35, species = "CO2")
gas_solubility(t = 0:5, S = 35, species = "O2")
Temp <- seq(from = 0, to = 30, by = 0.1)
mf <- par(mfrow = c(1, 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)

plot(Temp, gas_solubility(t = Temp, species = "CCl4"), xlab = "temperature",
     ylab = "mmol/m3/atm", main = "solubility (S=35)",
     type = "l", lwd = 2, ylim = c(0, 100000))
lines(Temp, gas_solubility(t = Temp, species = "CO2"), col = "red", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "N2O"), col = "blue", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "Rn"), col = "green", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "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, gas_solubility(t = Temp, species = "Kr"), xlab = "temperature",
     ylab = "mmol/m3/atm", main = "solubility (S=35)",
     type = "l", lwd = 2, ylim = c(0, 40000))
lines(Temp, gas_solubility(t = Temp, species = "CH4"), col = "red", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "Ar"), col = "blue", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "O2"), col = "green", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "N2"), col = "yellow", lwd = 2)
lines(Temp, gas_solubility(t = Temp, species = "Ne"), col = "grey", lwd = 2)
The Gas Transfer Coefficient in m/sec

Description

The gas transfer coefficient, in $ms^{-1}$, for certain gases in seawater ($S = 35$).
Usage

gas_transfer(t = 25, u10 = 1, species = c("He", "Ne", "N2", "O2", "Ar", "Kr", "Rn", "CH4", "CO2", "N2O", "CCl2F2", "CCL3F", "SF6", "CCl4"),
method = c("Liss", "Nightingale", "Wanninkhof1", "Wanninkhof2"),
Schmidt = gas_schmidt(t = t, species = species))

Arguments

t Temperature in °C,
u10 wind speed, in m/sec at a nominal height of 10 m above sea level,
species character vector with gasses whose gas transfer coefficient should be estimated.
Schmidt the Schmidt number, when given this overrules the arguments gas and t.

Value

The gas transfer velocity, for seawater, in m s^{-1}.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

gas_O2sat, gas_satconc, gas_schmidt, gas_solubility, atmComp, vapor
Examples

```r
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, 0.0003))
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)
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"))
```

Description

Computes the gravity, based on latitude.

Usage

```r
gravity(lat = 0, method = c("Moritz", "UNESCO"))
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lat</td>
<td>latitude (-90 - +90).</td>
</tr>
<tr>
<td>method</td>
<td>When &quot;UNESCO&quot;, uses the UNESCO (1983) polynomial, else according to Moritz, 2000</td>
</tr>
</tbody>
</table>

Value

Gravity, in m sec$^{-2}$.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References

The UNESCO polynomial:
http://unesdoc.unesco.org/images/0005/000598/059832EB.pdf

molvol

Mol to Liter Conversion for a Gas

Description

Converts from liter to moles for a gas.

Usage

```r
molvol(t = 25, P = 1.013253,
species = c("ideal", "Ar", "CO2", "CS2", "C0", "CC14", "Cl2",
"C3H8", "H2O", "He", "H2", "HBr", "HCL", "H2S", "Hg",
"Kr", "NH3", "Ne", "NO", "N2", "NO2", "N2O", "O2", "PH3",
"SiH4", "SiF4", "SO2", "Xe"),
quantity = 1, a = 0, b = 0)
```

Arguments

- `t` temperature, °C
- `P` True pressure, bar.
- `species` character vector with gasses whose molecular volume should be estimated. If `NULL` then the coefficients `a` and `b` are used.
- `quantity` mol of the gas.
- `a` Van der Waals constant a, a species-specific coefficient, \( dm^6 \times bar/mol^2 \).
- `b` Van der Waals constant b, a species-specific coefficient, \( dm^3/mol \).

Value

volume of the gas, liter

Note

The coefficients a and b are species-specific; values of 0 assume an ideal gas and in general give good estimates.

Use \( 1/molvol \) to convert from liter to moles.

The default calculates the molar volume of an ideal gas.

See Also

coriolis, viscosity, diffcoeff, ssd2rad, vertmean

Examples

```r
gravity(lat = 30)
```

```

molvol

Mol to Liter Conversion for a Gas

Description

Converts from liter to moles for a gas.

Usage

```r
molvol(t = 25, P = 1.013253,
species = c("ideal", "Ar", "CO2", "CS2", "C0", "CC14", "Cl2",
"C3H8", "H2O", "He", "H2", "HBr", "HCL", "H2S", "Hg",
"Kr", "NH3", "Ne", "NO", "N2", "NO2", "N2O", "O2", "PH3",
"SiH4", "SiF4", "SO2", "Xe"),
quantity = 1, a = 0, b = 0)
```

Arguments

- `t` temperature, °C
- `P` True pressure, bar.
- `species` character vector with gasses whose molecular volume should be estimated. If `NULL` then the coefficients `a` and `b` are used.
- `quantity` mol of the gas.
- `a` Van der Waals constant a, a species-specific coefficient, \( dm^6 \times bar/mol^2 \).
- `b` Van der Waals constant b, a species-specific coefficient, \( dm^3/mol \).

Value

volume of the gas, liter

Note

The coefficients a and b are species-specific; values of 0 assume an ideal gas and in general give good estimates.

Use \( 1/molvol \) to convert from liter to moles.

The default calculates the molar volume of an ideal gas.

See Also

coriolis, viscosity, diffcoeff, ssd2rad, vertmean

Examples

```r
gravity(lat = 30)
```
Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

References
The values of the van der Waals constants are from:
as found in: http://en.wikipedia.org/wiki/Van_der_Waals_constants_(data_page)

See Also
AtomicWeight, molweight, redfield

Examples
# molecular volume of an ideal gas.
molvol(species = "ideal", P = 1, t = 0) # 22.710 980
molvol(species = "ideal", P = 1, t = 25) # 24.789 598

plot(0:30, molvol(t = 0:30, species = NULL),
     xlab = "Temperature, dgC", ylab = "Molar volume")

#
molvol(a = 1.382, b = 0.03186, species = NULL, t = 0)
molvol(t = 0, species = "O2")

# the same but for all gasses
molvol(t = 0)

# table for different pressures
molvol(P = 1:5, species = "O2")

# the inverse function
1/molvol(species = "O2")

# contour plot
P <- seq(1, 100, by = 1)
Temp <- seq(-5, 40, by = 1)

Val <- outer(X = P, Y = Temp,
              FUN = function(X, Y) molvol(P = X, t = Y, species = "O2"))
contour(P, Temp, Val, xlab = "pressure", ylab = "temperature",
        main = "molvol", nlevel = 20, log = "x", axes = FALSE)
axis(1); axis(2); box()
molweight

Molecular Weight of a Chemical Species

Description

Calculates the molecular weight of chemical species.

Usage

molweight(species)

Arguments

species character vector with chemical species whose molecular weight is requested.

Details

Molecular weights of chemical elements may vary due to different isotope compositions, depending on geology, industrial processes or biological activity. Please consult the IUPAC Technical report about the details. The function returns NA for elements (and their compounds) which have no stable isotopes (except U, Th, Pa).

Value

Vector with the molecular weights in g/mol.

Note

This function uses text parsing of chemical formulae, it is strictly case sensitive.

Author(s)

Thomas Petzoldt

References


See Also

AtomicWeight, molvol, redfield
Examples

```r
molweight("CO2")
molweight("HCO3")
molweight("H")
molweight("H3PO4")

## eicosapentaenoic acid (EPA)
molweight("CH3CH2CHCH2CHCH2CHCH2CHCH2CHCH2CH(CH2)3COOH")
molweight("C20H30O2")

## works also with vectors
molweight(c("C2H5OH", "CO2", "H2O"))
molweight(c("SiOH4", "NaHCO3", "C6H12O6", "Ca(HCO3)2", "Pb(NO3)2", "(NH4)2SO4"))

## note that chemical formulae are case-sensitive
molweight("Co") # cobalt
molweight("CO") # carbon monoxide

## from gram to mol
1/molweight("CO3")
```

---

**Oceans**

*Useful Characteristics of the Oceans*

**Description**

Surface area and volume of the world’s oceans

**Usage**

Oceans

**Format**

A list specifying the value, units, and a description of each quantity.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


**See Also**

AtomicWeight, Bathymetry, Constants, earth_surf
Examples

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

redfield  

Redfield Ratio Calculator

Description

Estimate elemental composition of biomass (or media) according to the Redfield ratio.

Usage

redfield(q, species, method = c("mol", "mass"),
    ratio = c(C=106, H=263, O=110, N=16, P=1))

Arguments

q        amount of substance of that element (in mol or mass units),
species  The element that is given ("C", "H", "O", "N", "P"),
method   measurement unit ("mol" or "mass"),
ratio    average elemental composition.

Details

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, 1963, Richards 1965). Note that while the C:N:P ratio
is widely agreed there is still discussion about the average of O and H, e.g. $C_{106}H_{180}O_{45}N_{16}P_{1}$
(Stumm, 1964).

Note also that there are, of course, large differences depending on species and physiological state.

Value

A data frame with the estimated ratio of the main elements.

References

Redfield AC, 1934. On the proportions of organic derivations in sea water and their relation to the
Press of Liverpool, 177-192.

Redfield, AC, Ketchum, BH and Richards FA, 1963. The influence of organisms on the composition

Richards FA, 1965. Anoxic basins and fjords. In: Riley JP, Skirrow D. (Eds.), Chemical Oceanog-


See Also

AtomicWeight, molvol, molweight

Examples

```r
## Redfield ratio
redfield(1, "P")
## returns the molar Redfield ratio, rescaled to nitrogen
redfield(1, "N")
## how many mass units are related to 2 mass units (e.g. mg) P
redfield(2, "P", "mass")
redfield(c(1, 2, 3), "N", "mass")

## mass percentage of elements
x <- redfield(1, "P", "mass")
x / sum(x)

## with alternative elemental composition (Stumm, 1964)
x <- redfield(1, "P", "mass",
            ratio = c(C = 106, H = 180, O = 45, N = 16, P = 1))
x / sum(x)

## rule of thumb for fresh mass (in mg) formed by 1 microgram P
redfield(1, "P", "mass")$C * 2 * 10 / 1000
sum(redfield(1, "P", "mass",
            ratio = c(C = 106, H = 180, O = 45, N = 16, P = 1))) * 10 / 1000
```

### ssd2rad

#### Estimate Global Radiation from Measured Sunshine Duration Time

**Description**

The function converts values of sunshine duration (in hours) to global radiation (in $Jm^{-2}s^{-1}$).

**Usage**

```r
ssd2rad(s, doy, a = 0.25, b = 0.5, rho = 50.29)
```
Arguments

\( S \)  
Sunshine duration (hours)

\( \text{doy} \)  
Julian day (for northern hemisphere only)

\( a, b, \rho \)  
site specific conversion parameters, must be fitted to measured data.

Value

Estimated value of global radiation in \( J m^{-2} s^{-1} \).

Note

Don’t forget to fit the function parameters to site specific values!

References


See Also

coriolis, viscosity, diffcoeff, vertmean, gravity

Examples

```r
ssd2rad(8, 120)
```

---

**sw_adtgrad**

Adiabatic Temperature Gradient in Seawater

Description

Computes the adiabatic temperature gradient in seawater, using the UNESCO 1983 polynomial.

Also known as the adiabatic lapse rate, the change of temperature per unit pressure for an adiabatic change of pressure of an element of seawater. It is assumed that no heat or salt is exchanged with the surroundings.

Usage

```r
sw_adtgrad(S = 35, t = 25, p = P-1.013253, P = 1.013253 )
```

Arguments

\( S \)  
Practical salinity (-),

\( t \)  
Temperature, °C

\( p \)  
gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar

\( P \)  
true pressure, bar
**sw_alpha**

**Value**

adiabatic temperature gradient, in \( \text{dg K/ bar} \)

**Note**

Note: in the original formula, the units of `sw_adtgrad` are \( \text{dg K/\text{dbar}} \) (here: \( \text{dg K/\text{bar}} \)).

`sw_adtgrad` for \( S = 40, \ t = 40, p = 1000 \) is \( 3.255976e-3 \)

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


**See Also**

`sw_alpha`, `sw_beta`, `sw_comp`, `sw_conserv`, `sw_cp`, `sw_dens`, `sw_depth`, `sw_enthalpy`, `sw_entropy`, `sw_gibbs`, `sw_kappa`, `sw_kappa_t`, `sw_sfac`, `sw_svel`, `sw_tfreeze`, `sw_tpot`

**Examples**

```r
sw_adtgrad(t = 40, S = 40, p = 1000) # 3.255976e-4

## Check values
sw_adtgrad(S = 25, t = 10, p = 0)  # 0.1002e-3
sw_adtgrad(S = 25, t = 10, p = 100) # 0.1135e-3
sw_adtgrad(S = 25, t = 10, p = 1000) # 0.2869e-3

sw_adtgrad(S = 25, t = 30, p = 0)  # 0.2417e-3
sw_adtgrad(S = 40, t = 30, p = 0)  # 0.2510e-3
sw_adtgrad(S = 40, t = 0, p = 100) # 0.0630e-3
```

---

**sw_alpha**  
*Thermal Expansion Coefficient of Seawater*

**Description**

Computes the seawater thermal expansion coefficient with respect to in situ temperature, \( 1/\text{K} \)

**Usage**

```r
sw_alpha(S = 35, t = 25, p = P-1.013253, P = 1.013253)
```
Arguments

- **S**: Absolute salinity (g/kg).
- **t**: Temperature, °C,
- **p**: gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- **P**: true pressure, bar

Value

Thermal expansion coefficient, 1/K.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

- `sw_tfreeze`, `sw_tpot`, `sw_adtgrad`, `sw_beta`, `sw_comp`, `sw_conserv`, `sw_cp`, `sw_dens`, `sw_depth`,
- `sw_enthalpy`, `sw_entropy`, `sw_gibbs`, `sw_kappa`, `sw_kappa_t`, `sw_sfac`, `sw_svel`, `sw_tfreeze`,
- `sw_tpot`,
- `convert_PStoAS`, to convert from practical salinity (-) to absolute salinity (g/kg)

Examples

```r
sw_alpha(35.7, 25.5, 102.3) #0.000309837839319264
```

---

**sw_beta**  
*Haline Contraction Coefficient of Seawater*

Description

Computes the seawater haline contraction coefficient with respect to constant, in situ temperature, kg/g

Usage

```r
sw_beta(S = 35, t = 25, p = P-1.013253, P = 1.013253)
```
Arguments

- **S**: Absolute salinity (g/kg).
- **t**: Temperature, °C.
- **p**: Gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar.
- **p**: True pressure, bar.

Value

Haline contraction coefficient, kg/g.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

- `sw_adtgrad`, `sw_alpha`, `sw_comp`, `sw_conserv`, `sw_cp`, `sw_dens`, `sw_depth`, `sw_enthalpy`, `sw_entropy`, `sw_gibbs`, `sw_kappa`, `sw_kappa_t`, `sw_sfac`, `sw_svel`, `sw_tfreeze`, `sw_tpot`, `convert_PStoAS`, to convert from practical salinity (-) to absolute salinity (g/kg)

Examples

```r
sw_beta(35.7, 25.5, 102.3) # 0.00072572979797838666
```

---

sw_comp

**Reference Sea Salt Composition**

Description

The sea salt composition definition for reference salinity of the standard ocean at 25 °C and 1.01325 bar (atmospheric pressure), given in mass fractions.

Usage

```r
```
Arguments
species character vector with components whose composition should be estimated.

Value
A vector with the mass fractions.

Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

References

See Also
sw_adtgrad, sw_alpha, sw_beta, sw_conserv, sw_cp, sw_dens, sw_depth, swenthalpy, sw_entropy,
sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_freeze, sw_tpot

Examples
sw_comp("CO2")
sw_comp()
sum(sw_comp())

---

sw_conserv                  Concentrations of (Conservative) Species in Seawater

Description
Estimates the concentration of Borate, Calcite, Sulphate and Fluoride in seawater, as a function of
salinity.

Usage
sw_conserv(S = 35)

Arguments
S  Practical salinity, (-).

Details
The borate and calcite concentration as in Millero (1995),
Sulphate as in Morris and Riley, 1966,
Fluoride as in Riley, 1965.
sw_cp

Value
A data frame with the concentrations in micromol/kg.

Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

References

See Also
sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot

Examples
data.frame(salinity = 1:35, sw_conserv(1:35))

sw_cp

Heat Capacity of Sea Water

Description
Estimates the heat capacity of seawater.
Valid for S = 0 to 40, T = 0 to 35 °C

Usage
sw_cp(S = 35, t = 25, p = P-1.013253, P = 1.013253, method = c("Gibbs", "UNESCO"))

Arguments
S Salinity, when method = "UNESCO": practical salinity (-) else absolute salinity (g/kg).
t Temperature, °C,
p gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
P true pressure, bar
method When "UNESCO", uses the UNESCO (1983) polynomial, when "Gibbs", based on the gibbs functions as in Feistel, 2008
Value

Heat capacity, in \( Jkg^{-1}dgC^{-1} \)

Note

\( p \) is applied pressure, 0 bar at sea surface.

when using UNESCO polynomial, \( cp \) for \( S = 40, T = 40, P = 1000 \) is 3849.5 J/(kg dg C).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

\( \text{sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot} \)

convert_PStoAS, to convert from practical salinity (-) to absolute salinity (g/kg)

convert_AStoPS, to convert from absolute salinity (g/kg) to practical salinity (-)

Examples

\[
\text{sw_cp}(S = 40, t = 40, p = 1000, \text{method} = "UNESCO") \quad # \ 3849.5
\]

# Check value Gibbs function
\[
\text{sw_cp}(35.7,25.5,102.3)\#93974.42541259729
\]

# Check values UNESCO
\[
\text{sw_cp}(S = 25, t = 10, p = 0, \text{method} = "UNESCO") \quad # \ 4041.8
\]
\[
\text{sw_cp}(S = 25, t = 10, p = 1000, \text{method} = "UNESCO") \quad # \ 3842.3
\]
\[
\text{sw_cp}(S = 25, t = 30, p = 0, \text{method} = "UNESCO") \quad # \ 4049.1
\]
\[
\text{sw_cp}(S = 40, t = 10, p = 0, \text{method} = "UNESCO") \quad # \ 3959.3
\]
**sw_dens**

*Density of Sea Water*

### Description

Density of sea water in $kgm^{-3}$

### Usage

```
sw_dens(S = 35, t = 25, p = max(0, P-1.013253), P = 1.013253,
method=c("Gibbs","UNESCO","Chen"))
```

### Arguments

- **S**: Salinity, when method = "UNESCO": practical salinity (-) else absolute salinity (g/kg).
- **t**: Temperature, °C.
- **p**: gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- **P**: true pressure, bar
- **method**: When "UNESCO", uses the UNESCO (1983) polynomial, when "Gibbs", based on the Gibbs functions as in Feistel, 2008 "Chen" for the limnological range (i.e. fresh water systems).

### Details

To avoid confusion between the arguments (S, t, p) it is advisable to use named arguments in general (e.g. `rho(t = 4)`). The UNESCO formula is imported from package `seacarb`.

### Value

Density of water in $kgm^{-3}$.

### Note

Pressure used here is 1 bar (true pressure), in contrast to hydrostatic pressure (0 bar at surface) in original formula.

The coefficients from McDougall et al., 2009 were used. For temperature, they differ slightly from Feistel 2003 and Feistel 2008, which is why, for temperatures different from 0, there is a slight offset from the estimates as from table 22 or 21 from Feistel (2008).
References


See Also

rho in package seacarb.

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot

convert_PStoAS, to convert from practical salinity to absolute salinity

convert_AStoPS, to convert from absolute salinity to practical salinity

Examples

# table 22 Feistel 2008
sw_dens(0, 0, 0) #0.999843086e3
sw_dens(0, 79.85, 0) #0.97188383e3 - deviates
sw_dens(0, 0, 998.98675) #0.104527796e4

# table 21 Feistel 2008
sw_dens(35.16504, 0, 0) #0.10281072e4
sw_dens(100, 79.85, 0) #0.102985888e4
sw_dens(35.16504, 0, 998.98675) #0.10709264e4

sw_dens(35.7, 25.5, 102.3) #1027.95249315662

S <- 0:40
plot(S, sw_dens(S = S, t = 4, method = "UNESCO"))

lines(S, sw_dens(S = S, t = 4, method = "Gibbs"), col = "red")

lines(S, sw_dens(S = S, t = 4, method = "Chen"), col = "blue")
**Description**

 Computes the water depth for water of salinity 35, and temperature 0 \( \text{dg C} \), based on latitude and hydrostatic pressure, using the UNESCO 1983 polynomial.

**Usage**

\[
sw\_depth(p = P - 1.013253, \ P = 1.013253, \ \text{lat} = 0)
\]

**Arguments**

- \( p \)  
  gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- \( P \)  
  true pressure, bar
- \( \text{lat} \)  
  latitude (-90 to +90), -

**Value**

Water depth in m.

**Note**

\( sw\_depth \) for \( p = 1000, \ \text{lat} = 30 \) is 9712.653 m.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**

http://unesdoc.unesco.org/images/0005/000598/059832EB.pdf

**See Also**

\( sw\_adtgrad, sw\_alpha, sw\_beta, sw\_comp, sw\_conserv, sw\_cp, sw\_dens, sw\_enthalpy, sw\_entropy, \\
sw\_gibbs, sw\_kappa, sw\_kappa\_t, sw\_sfac, sw\_svel, sw\_tfreeze, sw\_tpot \)
Examples

\begin{verbatim}
sw_depth(p = 1000, lat = 30:40)

## Check values
sw_depth(p = 1000, lat = 30)  #9712.65
sw_depth(p = 50, lat = 30)   #496.00
sw_depth(p = 50, lat = 60)   #494.69
sw_depth(p = 500, lat = 60)  #4895.60
\end{verbatim}

\section*{sw_enthalpy

\textit{Specific Enthalpy of Seawater}}

\section*{Description}

Computes the seawater specific enthalpy, J/kg

\section*{Usage}

\begin{verbatim}
sw_enthalpy(S = 35, t = 25, p = P-1.013253, P = 1.013253)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \textbf{S} Absolute salinity (g/kg),
  \item \textbf{t} Temperature, °C,
  \item \textbf{p} gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
  \item \textbf{P} true pressure, bar
\end{itemize}

\section*{Value}

Specific enthalpy, J/kg.

\section*{Author(s)}

Karline Soetaert <karline.soetaert@nioz.nl>

\section*{References}


See Also

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot
convert_PStoAS, to convert from practical salinity (-) to absolute salinity (g/kg)

Examples

sw_enthalpy(35.7, 25.5, 102.3) #110776.712408975

sw_entropy

Specific Entropy of Seawater

Description

Computes the seawater specific entropy, J/(kg*K)

Usage

sw_entropy(S = 35, t = 25, p = P-1.013253, P = 1.013253)

Arguments

S  Absolute salinity (g/kg),
t  Temperature, °C,
p  gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
P  true pressure, bar

Value

Specific entropy, J/(kg*K).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


sw_gibbs

See Also
sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot, convert_PStoAS, to convert from practical salinity (-) to absolute salinity (g/kg)

Examples
sw_entropy(35.7, 25.5, 102.3) #352.81879771528

---

**Gibbs Function of Seawater**

**Description**
Calculates the seawater specific gibbs free energy, including derivatives up to order 2, for a given temperature, salinity and pressure.

The Gibbs function of seawater \( g(S, t, p) \) is related to the specific enthalpy \( h \) and entropy \( s \), by \( g = h - (273.15 \text{ K} + t) s \)

**Usage**

```
sw_gibbs(S = 35, t = 25, p = P-1.013253,
P = 1.013253, dS = 0, dt = 0, dp = 0)
```

**Arguments**

- **S**: Absolute salinity (g/kg),
- **t**: Temperature, °C,
- **p**: gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- **P**: true pressure, bar
- **dS**: order of the S derivative
- **dt**: order of the t derivative
- **dp**: order of the p derivative

**Value**
The Gibbs function, J/kg, or its derivative

**Note**
The gibbs function is defined as the sum of a pure water part and the saline part (IAPWS-08)
The coefficients from McDougall et al., 2009 were used. For temperature, they differ slightly from Feistel 2003 and Feistel 2008, which is why, for temperatures different from 0, there is a slight offset from the estimates as from table 22 or 21 from Feistel (2008).
**sw_gibbs**

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


**See Also**

- `sw_adtgrad`, `sw_alpha`, `sw_beta`, `sw_comp`, `sw_conserv`, `sw_cp`, `sw_dens`, `sw_depth`, `sw_enthalpy`, `sw_entropy`, `sw_kappa`, `sw_kappa_t`, `sw_sfac`, `sw_svel`, `sw_tfreeze`, `sw_tpot`
- `convert_PStoAS`, to convert from practical salinity (-) to absolute salinity (g/kg)

**Examples**

```r
# table 22 Feistel 2008
sw_gibbs(0, 0, 0)  # = 101.34274
sw_gibbs(0, 0, 0, dS = 1)  # 0
sw_gibbs(0, 0, 0, dt = 1)  # 0.14764376
sw_gibbs(0, 0, 0, dp = 1)  # 0.1000015694e-2
sw_gibbs(0, 0, 0, dS = 1, dp = 1)  # 0
sw_gibbs(0, 0, 0, dt = 1, dp = 1)  # 0.677700318e-7

sw_gibbs(0, 79.85, 0)  # -0.446114969e5 differs (see note)
sw_gibbs(0, 79.85, 0, dt = 1)  # -0.107375993e4 differs
sw_gibbs(0, 79.85, 0, dp = 1)  # 0.102892956e-2 differs
sw_gibbs(0, 79.85, 0, dS = 1, dp = 1)  # 0
sw_gibbs(0, 79.85, 0, dt = 1, dp = 1)  # 0.659051552e-6

sw_gibbs(0, 0, 998.98675)  # 0.977303862e5
sw_gibbs(0, 0, 998.98675, dt = 1)  # 0.851466502e1
sw_gibbs(0, 0, 998.98675, dp = 1)  # 0.95683329e-3
sw_gibbs(0, 0, 998.98675, dS = 1, dp = 1)  # 0
sw_gibbs(0, 0, 998.98675, dt = 1, dp = 1)  # 0.199079571e-6

# table 21 Feistel 2008
sw_gibbs(35.16504, 0, 0)  # = 0
sw_gibbs(35.16504, 0, 0, dS = 1)  # 0.639974067e2 differs
sw_gibbs(35.16504, 0, 0, dt = 1)  # 0
sw_gibbs(35.16504, 0, 0, dp = 1)  # 0.972661217e-3
sw_gibbs(35.16504, 0, 0, dS = 1, dp = 1)  # 0.759615412e-6
sw_gibbs(35.16504, 0, 0, dt = 1, dp = 1)  # 0.515167556e-7 !!!

sw_gibbs(100, 79.85, 0)  # -0.295243229e5 differs
sw_gibbs(100, 79.85, 0, dS = 1)  # 0.251957276e3
sw_gibbs(100, 79.85, 0, dt = 1)  # -0.917529024e3 differs
```
`sw_kappa`  

Isentropic Compressibility of Seawater

**Description**

Computes the seawater isentropic compressibility, 1/bar

**Usage**

```r
sw_kappa(S = 35, t = 25, p = P-1.013253, P = 1.013253)
```

**Arguments**

- `S`  
  Salinity (dimensionless),

- `t`  
  Temperature, °C,

- `p`  
  gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar

- `P`  
  true pressure, bar

**Value**

Isentropic compressibility, 1/bar

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


**sw_kappa_t**

**See Also**

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze, sw_tpot

**convert_PStoAS**, to convert from practical salinity (-) to absolute salinity (g/kg)

**Examples**

```r
sw_kappa(35.7, 25.5, 102.3) #4.03386268546478e-6
```

---

**sw_kappa_t**  
*Isothermal Compressibility of Seawater*

**Description**

Computes the seawater isothermal compressibility, 1/Pa

**Usage**

```r
sw_kappa_t(S = 35, t = 25, p = P-1.013253, P = 1.013253)
```

**Arguments**

- **S** Absolute salinity (g/kg),
- **t** Temperature, °C,
- **p** gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- **P** true pressure, bar

**Value**

isothermal compressibility, 1/Pa.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


See Also

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_sfac, sw_svel, sw_tfreeze, sw_tpot

calculate_PStoAS, to convert from practical salinity (−) to absolute salinity (g/kg)

Examples

sw_kappa_t(35.7, 25.5, 102.3) #4.10403794615135e-6

---

**sw_sfac**

*Salinity conversion factors*

---

**Description**

Factors to convert from practical to absolute salinity and vice versa.

**Usage**

sw_sfac

**Format**

A list with the following:

- **longs** the longitude, a vector with 91 elements, range (0,360), third dimension in del_sa,
- **lats** the latitude, second dimension in del_sa, a vector with 44 elements, range (-82,90),
- **p** dbar, the first dimension in del_sa, a vector with 45 elements, range(0,6131),
- **ndepth** the number of depth intervals at a certain lat,long, a matrix of dimension (4,91),
- **del_sa** the salinity anomaly, an array with dimension (45,44,91), i.e. for (p, lats, longs) values.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


sw_svel

**Velocity of the Sound in Seawater**

**Description**

Computes the velocity of the sound in seawater, using the UNESCO 1983 polynomial or based on the Gibbs function.

Valid for salinity from 0 to 40, temperature from 0 to 40 °C, pressure from 1 to 1000 bars.

**Usage**

```r
sw_svel(S = 35, t = 25, p = P - 1.013253, P = 1.013253,
method = c("Gibbs", "UNESCO"))
```

**Arguments**

- **S**  
  Salinity, when method = "UNESCO": practical salinity (g/kg), else absolute salinity (g/kg).
- **t**  
  Temperature, °C,
- **p**  
  gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
P  true pressure, bar
method  When "UNESCO", uses the UNESCO (1983) polynomial, when "Gibbs", based on the gibbs functions as in Feistel, 2008

Value

Sound velocity, in m / sec.

Note

Sound velocity for S = 40, t = 40, p = 1000 is 1731.995 using UNESCO polynomial.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_tfreeze, sw_tpot
convert_PStoAS, to convert from practical salinity (-) to absolute salinity (g/kg)
convert_AStoPS, to convert from absolute salinity (g/kg) to practical salinity (-)

Examples

sw_svel(t = 40, S = 40, p = 10:20, method = "UNESCO")

# Check value UNESCO
sw_svel(t = 40, S = 40, p = 1000, method = "UNESCO")  # 1731.995
sw_svel(t = 0, S = 40, p = 0, method = "UNESCO")  # 1455.8

sw_svel(t = 40, S = 25, p = 1000, method = "UNESCO")  # 1719.2
sw_svel(t = 40, S = 25, p = 0, method = "UNESCO")  # 1553.4
sw_svel(t = 0, S = 25, p = 0, method = "UNESCO")  # 1435.8

# Check value Gibbs
sw_svel(S = 35.7, t = 25.5, p = 102.3)  # 1552.93372863425
**sw_tfreeze**

*Freezing Temperature of Seawater*

**Description**
Estimates the freezing temperature of seawater, using the UNESCO 1983 polynomial.
Valid for salinity 4-40

**Usage**

\[
\text{sw_tfreeze}(S=35, \ p = P\cdot 1.013253, \ P = 1.013253 )
\]

**Arguments**

- **S**  
  practical salinity, \(\text{-}\).
- **p**  
  gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- **P**  
  true pressure, bar

**Value**

Temperature, \(\circ\)C

**Note**

freezing temperature for \(S = 40, \ p = 50\) is -2.588567 \(\circ\)C.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**

http://unesdoc.unesco.org/images/0005/000598/059832EB.pdf

**See Also**

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tpot
Examples

```r
sw_tfreeze(S = 40, p = 50)
```

```r
## Check values
sw_tfreeze(S = 10, p = 0) # -0.542
sw_tfreeze(S = 10, p = 10) # -0.618
sw_tfreeze(S = 30, p = 0) # -1.638
sw_tfreeze(S = 40, p = 50) # -2.589
```

---

**sw_tpot**  
*Potential Temperature of Seawater*

**Description**

Estimates the potential temperature of seawater, using the UNESCO 1983 polynomial. It is the temperature an element of seawater would have if raised adiabatically with no change of salinity, to atmospheric pressure.

**Usage**

```r
sw_tpot(S = 35, t = 25, p, pref = 0)
```

**Arguments**

- `t`  
  temperature, °C,
- `S`  
  practical salinity, -,
- `p`  
  gauge or applied pressure, pressure referenced against the local atmospheric pressure, bar
- `pref`  
  reference hydrostatic pressure, bar.

**Value**

Temperature, °C.

**Note**

`sw_tpot` for `S = 40, t = 40, p = 1000` is 36.89073 dgC

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**References**


vapor

See Also

sw_adtgrad, sw_alpha, sw_beta, sw_comp, sw_conserv, sw_cp, sw_dens, sw_depth, sw_enthalpy, sw_entropy, sw_gibbs, sw_kappa, sw_kappa_t, sw_sfac, sw_svel, sw_tfreeze

Examples

sw_tpot(S = 40, t = 40:45, p = 1000)

## check values
sw_tpot(S = 25, t = 40, p = 0) #40
sw_tpot(S = 25, t = 40, p = 100) #36.6921
sw_tpot(S = 25, t = 10, p = 1000) #8.4684
sw_tpot(S = 25, t = 0, p = 1000) #0.0265
sw_tpot(S = 40, t = 40, p = 1000) #36.89073

---

vapor Saturation Water Vapor Pressure

Description

The partial pressure of water in saturated air (pH20/P), as in Weiss and Price (1980), where P is the total atmospheric pressure, (1 atmosphere), and pH2O is the partial pressure of the water vapor.

Usage

vapor(S = 35, t = 25)

Arguments

S Salinity (-),
t Temperature, °C.

Value

The saturation vapor pressure (-).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


**See Also**

gas_O2sat, gas_satconc, gas_schmidt, gas_solubility, gas_transfer, atmComp

**Examples**

```r
plot(0:30, vapor.hPa(t = 0:30), xlab = "Temperature, dgC", ylab = "pH2O/P")
```

---

**vapor.hPa**

**Vapor Pressure**

**Description**

The vapor pressure of water, in hPa.

**Usage**

```r
vapor.hPa(t = 25)
```

**Arguments**

- **t**
  - Temperature, °C.

**Value**

The vapor pressure of water, in hecto Pascal; valid for temperature of [-50,100] dgC.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>, Lorenz Meire <lorenz.meire@nioz.nl>

**References**


http://www.cactus2000.de/uk/unit/masshum.shtml

**See Also**

vapor, air_spechum, air_density, gas_O2sat, gas_satconc, gas_schmidt, gas_solubility, gas_transfer, atmComp

**Examples**

```r
vapor.hPa(t = 25)
plot(0:30, vapor.hPa(t = 0:30), xlab = "Temperature, dgC", ylab = "hPa")
```
Description

Calculate vertical mean values which respect to depths of different layers or lake morphometry.

Usage

vertmean(depth, vari, level, top, bot, vol, total=FALSE)

Arguments

depth sorted vector of sampled depths,
vari measurements corresponding to depth (concentration, temperature, . . . ),
level surface water level (above ground or above sea level (m a.s.l.), depending on bathymetric function used,
top top water level of requested layer over which to average or integrate,
bot bottom water level of requested layer over which to average or integrate,
vol hypsographic function to be used (e.g. vol.depth),
total if TRUE the total sum over the water body is returned (integrated value), instead of the volumetric mean.

Value

Volumetric average respectively total value (for total =TRUE) for a given quantity (concentration, energy, temperature) in the requested layer between depths top and bottom.

Author(s)

Thomas Petzoldt

See Also

coriolis, viscosity, diffcoeff, ssd2rad, gravity

Examples

## define a bathymetric formula for a given lake or basin
## z: water depth (m below surface)
## zz: water column (m above ground)
## level: total water depth (m above ground or above reference level)
weight.vol <- function(z, level) {
  zz <- level - z
  if (any(zz < 0)) stop(“depth > maximum depth”)
vol <- 175947 * zz^2 + 2686 * zz^3 # m^3
}

## area is first derivative
area <- function(z, level) {
  zz <- level - z
  A <- 0.5 * 175947 * zz + 1/3 * 2686 * zz^2 # m^2
}

## dummy formula for depth-weighted averaging
## (water column, instead of bathymetric curve)
weight.column <- function(z, level) {z}

## Plot of lake volume (bathymetric curve)
par(mfrow = c(1, 2))
z <- 0:12
V <- weight.vol(z, 12)
plot(V, z, type = "l", ylim = c(12, 0), xlab = "Volume (m3)", ylab = "Depth (m)"
    polygon(c(V, 0), c(z, 0), col = "cyan")

## Test Data
level <- 12
depth <- c(0, 1, 3.5, 5, 7, 10, 10.5, 11.5)
pconc <- c(3.7, 4.2, 6.1, 8.9, 7.8, 9.7, 11.4, 11.4)

## Plot test data
plot(pconc, depth, xlim=range(c(0, pconc)), ylim=c(12,0), type="n",
    xlab="P concentration (mu g / L)", ylab="Depth (m)"
    segments(rep(0, 13), depth, pconc, depth, lwd=3)

## simple means
m <- mean(pconc[depth <= 4])
lines(c(m, m), c(0, 4), col="blue", lwd=2)

m <- mean(pconc[depth >= 4])
lines(c(m, m), c(4, 12), col="blue", lwd=2)

## depth weighted
m <- vertmean(depth, pconc, level, top=0, bot=4, weight.column)
lines(c(m, m), c(0, 4), col="red", lwd=2)

m <- vertmean(depth, pconc, level, top=4, bot=12, weight.column)
lines(c(m, m), c(4, 12), col="red", lwd=2)

## volume weighted
m <- vertmean(depth, pconc, level, top=0, bot=4, weight.vol)
lines(c(m, m), c(0, 4), col="green", lwd=2)

m <- vertmean(depth, pconc, level, top=4, bot=12, weight.vol)
lines(c(m, m), c(4, 12), col="green", lwd=2)

m <- vertmean(depth, pconc, level, top=4, bot=12, weight.vol)
Shear Viscosity of Water

Description

Calculates the shear viscosity of water, in centipoise (g/m/sec). Valid for 0 < t < 30 °C, 0 < S < 36, 1 < P < 1000 bars.

Based on the code "CANDI" by B.P. Boudreau

Usage

viscosity(S = 35, t = 25, P = 1.013253)

Arguments

S  salinity, -,

 t  temperature, °C,

 P  True pressure, bar.

Details

The details given in the original code by B. Boudreau are repeated here:

Uses the equation given by Kukulka et al. (1987).

Value

Shear viscosity in centipoise.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References

Based on the FORTRAN implementation of the diagenetic model "CANDI" of B.P. Boudreau:


See Also
coriolis, diffcoeff, ssd2rad, vertmean, gravity

Examples

plot(0:30, viscosity(t = 0:30, S = 35, P = 1),
     xlab = "temperature", ylab = "g/m/s",
     main = "shear viscosity of water", type = "l")
lines(0:30, viscosity(t = 0:30, S = 0, P = 1), col = "red")
lines(0:30, viscosity(t = 0:30, S = 35, 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"))
Index

*Topic datasets
  AtomicWeight, 8
  Bathymetry, 9
  Constants, 10
  Oceans, 37
  sw_sfac, 56
*Topic misc
  convert_p, 11
  convert_T, 17
  gas_02sat, 23
  redfield, 38
  ssd2rad, 39
  sw_dens, 47
  vertmean, 63
*Topic package
  marelac-package, 3
*Topic utilities
  air_density, 5
  air_spechum, 6
  atmComp, 7
  convert_RtoS, 11
  convert_salinity, 13
  convert_StoCl, 15
  convert_StoR, 16
  coriolis, 18
  diffcoeff, 19
  earth_surf, 21
  gas_satconc, 25
  gas_schmidt, 27
  gas_solubility, 28
  gas_transfer, 31
  gravity, 33
  molvol, 34
  molweight, 36
  sw_adtgrad, 40
  sw_alpha, 41
  sw_beta, 42
  sw_comp, 43
  sw_conserv, 44
  sw_cp, 45
  sw_depth, 49
  sw_enthalpy, 50
  sw_entropy, 51
  sw_gibbs, 52
  sw_kappa, 54
  sw_kappa_t, 55
  sw_svel, 57
  sw_tfreeze, 59
  sw_tpot, 60
  vapor, 61
  vapor.hPa, 62
  viscosity, 65

air_density, 4, 5, 6, 62
air_spechum, 4, 5, 6, 62
atmComp, 4–6, 7, 26, 28, 30, 32, 62
AtomicWeight, 4, 8, 10, 35–37, 39
atomicweight (AtomicWeight), 8

Bathymetry, 4, 8, 9, 10, 22, 37

Constants, 4, 8, 10, 37
convert_AStoPS, 4, 11, 12, 15–17, 46, 48, 57, 58
convert_AStoPS (convert_salinity), 13
convert_p, 4, 11, 12, 14–17
convert_PStoAS, 4, 11, 12, 15–17, 42, 43, 46, 48, 51–53, 55–58
convert_PStoAS (convert_salinity), 13
convert_RtoS, 4, 11, 14–17
convert_salinity, 13
convert_StoCl, 4, 11, 12, 14, 15, 16, 17
convert_StoR, 4, 11, 12, 14, 15, 16, 17
convert_T, 4, 11, 12, 14–16, 17
coriolis, 4, 18, 21, 34, 40, 63, 66
diffcoeff, 4, 18, 19, 34, 40, 63, 66
earth_dist (earth_surf), 21
earth_surf, 4, 21, 37
expand.grid, 20

gas_O2sat, 4–7, 23, 26, 28, 30, 32, 62
gas_satconc, 4–7, 24, 25, 28, 30, 32, 62
gas_schmidt, 4–7, 26, 27, 30, 32, 62
gas_solubility, 4–7, 26, 28, 28, 32, 62
gas_transfer, 4–7, 26, 28, 30, 31, 62
gravity, 4, 18, 21, 33, 40, 63, 66

marelac (marelac-package), 3
marelac-package, 3
molvol, 4, 34, 36, 39
molweight, 4, 8, 35, 36, 39

Oceans, 4, 8, 10, 22, 37

redfield, 4, 35, 36, 38
rho, 48

ssd2rad, 4, 18, 21, 34, 39, 63, 66
sw_adtgrad, 4, 40, 42–46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_alpha, 4, 41, 41, 43–46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_beta, 4, 41, 42, 44–46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_comp, 4, 41–43, 43, 45, 46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_conserv, 4, 41–44, 44, 46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_cp, 4, 41–45, 45, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_dens, 4, 41–46, 47, 49, 51–53, 55, 56, 58, 59, 61
sw_depth, 4, 41–46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_enthalpy, 4, 41–46, 48, 49, 50, 52, 53, 55, 56, 58, 59, 61
sw_entropy, 4, 41–46, 48, 49, 51, 51, 53, 55, 56, 58, 59, 61
sw_gibbs, 4, 41–46, 48, 49, 51, 52, 52, 55, 56, 58, 59, 61
sw_kappa_t, 4, 41–46, 48, 49, 51–53, 55, 55, 58, 59, 61
sw_sfac, 4, 41–46, 48, 49, 51–53, 55, 56, 56, 58, 59, 61
sw_svel, 4, 41–46, 48, 49, 51–53, 55, 56, 57, 59, 61
sw_tfreeze, 4, 41–46, 48, 49, 51–53, 55, 56, 58, 59, 61
sw_tpot, 4, 41–46, 48, 49, 51–53, 55, 56, 58, 59, 60
vapor, 4–7, 26, 28, 30, 32, 61, 62
vapor.hPa, 62
vertmean, 4, 18, 21, 34, 40, 63, 66
viscosity, 4, 18, 21, 34, 40, 63, 65