Package 'CHNOSZ'

February 11, 2024

Date 2024-02-11

Version 2.1.0

Title Thermodynamic Calculations and Diagrams for Geochemistry

Author Jeffrey Dick [aut, cre] (0000-0002-0687-5890)

Maintainer Jeffrey Dick <j3ffdick@gmail.com>

Depends R (>= 3.1.0)

Suggests tinytest, knitr, rmarkdown, tufte

Imports grDevices, graphics, stats, utils

Description An integrated set of tools for thermodynamic calculations in aqueous geochemistry and geobiochemistry. Functions are provided for writing balanced reactions to form species from user-selected basis species and for calculating the standard molal properties of species and reactions, including the standard Gibbs energy and equilibrium constant. Calculations of the non-equilibrium chemical affinity and equilibrium chemical activity of species can be portrayed on diagrams as a function of temperature, pressure, or activity of basis species; in two dimensions, this gives a maximum affinity or predominance diagram. The diagrams have formatted chemical formulas and axis labels, and water stability limits can be added to Eh-pH, oxygen fugacity-temperature, and other diagrams with a redox variable. The package has been developed to handle common calculations in aqueous geochemistry, such as solubility due to complexation of metal ions, mineral buffers of redox or pH, and changing the basis species across a diagram (``mosaic diagrams''). CHNOSZ also implements a group additivity algorithm for the standard thermodynamic properties of proteins.

Encoding UTF-8

License GPL-3

VignetteBuilder knitr

URL https://www.chnosz.net/, https://r-forge.r-project.org/projects/chnosz/

R topics documented:

`CHNOSZ-package` *Thermodynamic Calculations and Diagrams for Geochemistry*

Description

CHNOSZ is a package for thermodynamic calculations, primarily with applications in geochemistry and compositional biology. It can be used to calculate the standard molal thermodynamic properties and chemical affinities of reactions relevant to geobiochemical processes, and to visualize the equilibrium activities of species on chemical speciation and predominance diagrams.

Warm Tips

- To view the manual, run `help.start()` then select 'Packages' and 'CHNOSZ'. Examples in the function help pages can be run by pasting the code block into the R console.
- Also check out the vignette **anintro.html** (*An Introduction to CHNOSZ*).
- Run the command `examples()` to run all of the examples provided in CHNOSZ. This should take about a minute.

Getting Help

Each help page (other than this one) has been given one of the following "concept index entries":

- Main workflow: `info`, `subcrt`, `basis`, `species`, `affinity`, `equilibrate`, `diagram`
- Extended workflow: `swap.basis`, `buffer`, `mosaic`, `EOSregress`
- Thermodynamic data: `data`, `extdata`, `add.OBIGT`, `util.data`
- Thermodynamic calculations: `util.formula`, `makeup`, `util.units`, `Berman`, `nonideal`, `util.misc`
- Water properties: `water`, `util.water`, `DEW`, `IAPWS95`
- Protein properties: `protein.info`, `add.protein`, `util.fasta`, `util.protein`, `util.seq`, `ionize.aa`
- Other tools: `examples`, `taxonomy`
- Utility functions: `util.expression`, `util.plot`, `util.array`, `util.list`, `palply`

These concept entries are visible to `help.search` (aka `??`). For example, help pages related to thermodynamic data can be listed using `??"thermodynamic data"`.

Warning

All thermodynamic data and examples are provided on an as-is basis. It is up to you to check not only the accuracy of the data, but also the *suitability of the data AND computational techniques* for your problem. By combining data taken from different sources, it is possible to build an inconsistent and/or nonsensical calculation. An attempt has been made to provide a default database (OBIGT) that is internally consistent, but no guarantee can be made. If there is any doubt about the accuracy or suitability of data for a particular problem, please consult the primary sources (see `thermo.refs`).

Acknowledgements

This package would not exist without the scientific influence and friendship of the late Professor Harold C. Helgeson. The 'src/H2O92D.f' file with Fortran code for calculating the thermodynamic and electrostatic properties of H_2O is modified from the SUPCRT92 package (Johnson et al., 1992).

Work on CHNOSZ at U.C. Berkeley from ca. 2003 to 2008 was supported in part by research grants to HCH from the U.S. National Science Foundation and Department of Energy. In 2009–2011, development of this package was partially supported by NSF grant EAR-0847616 to JMD.

References

Johnson, J. W., Oelkers, E. H. and Helgeson, H. C. (1992) SUPCRT92: A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bar and 0 to 1000°C. *Comp. Geosci.* **18**, 899–947. doi:10.1016/0098-3004(92)90029Q

---

add.OBIGT *Functions to Work with the Thermodynamic Database*

Description

Add or modify species in the thermodynamic database.

Usage

```
add.OBIGT(file, species = NULL, force = TRUE)
mod.OBIGT(..., zap = FALSE)
```

Arguments

file	character, path to a file
species	character, names of species to load from file
force	logical, force replacement of already existing species?
...	character or numeric, properties of species to modify in the thermodynamic database
zap	logical, clear preexisting parameters?

Details

Note: change made to OBIGT are lost if you reload the database by calling reset or OBIGT or if you quit the R session without saving it.

add.OBIGT is used to update the thermodynamic database (thermo$OBIGT) in the running session. The format (column names) of the specified file must be the same as the extdata/OBIGT/*.csv files provided with CHNOSZ. Note that this includes both the E_units and model columns, which were added in versions 1.3.3 and 2.0.0.

file is first matched against the names of files in the extdata/OBIGT directory packaged with CHNOSZ. In this case, the file suffixes are removed, so 'DEW', 'organic_aq', and 'organic_cr' are valid names. If there are no matches to a system file, then file is interpreted as the path a user-supplied file.

If species is NULL (default), all species listed in the file are used. If species is given and matches the name(s) of species in the file, only those species are added to the database.

By default, species in the file replace any existing species having the same combination of name and state. Set force to FALSE to avoid replacing species that are present in (thermo()$OBIGT).

When adding (not replacing) species, there is no attempt made to keep the order of physical states in the database (aq-cr-liq-gas); the function simply adds new rows to the end of thermo$OBIGT. As a result, retrieving the properties of an added aqueous species using info requires an explicit state="aq" argument to that function if a species with the same name is present in one of the cr, liq or gas states.

mod.OBIGT changes one or more of the properties of species or adds species to the thermodynamic database. The name of the species to add or change must be supplied as the first argument of ... or as a named argument (named 'name'). Additional arguments to mod.OBIGT refer to the name of the property(s) to be updated and are matched to any part of compound column names in thermo()$OBIGT. For instance, either 'z' or 'T' matches the 'z.T' column. The values provided should also include order-of-magnitude scaling of HKF and DEW model parameters (see thermo).

When adding new species, a chemical formula should be included along with the values of any of the thermodynamic properties. The formula is taken from the 'formula' argument, or if that is missing, is taken to be the same as the 'name' of the species. An error occurs if the formula is not valid (i.e. can not be parsed by makeup). For new species, properties that are not specified become NA, except for 'state' and 'E_units', which take default values from thermo()$opt. These defaults can be overridden by giving a value for 'state' or 'E_units' in the arguments.

'model', if missing, is set to 'HKF' for state == "aq" or 'CGL' otherwise. When modifying some existing minerals in OBIGT, model = "CGL" should be explicitly given in order to override the Berman model.

When modifying species, the parameters indicated by the named arguments of mod.OBIGT are updated. Use zap = TRUE to replace all prexisting parameters (except for state and model) with NA values.

Value

The values returned (invisible-y) are the indices of the added and/or modified species.

References

Apps, J. and Spycher, N. (2004) *Data qualification for thermodynamic data used to support THC calculations*. DOC.20041118.0004 ANL-NBS-HS-000043 REV 00. Bechtel SAIC Company, LLC.

Bazarkina, E. F., Zotov, A. V., and Akinfiev, N. N. (2010) Pressure-dependent stability of cadmium chloride complexes: Potentiometric measurements at 1-1000 bar and 25°C. *Geology of Ore Deposits* **52**, 167–178. doi:10.1134/S1075701510020054

Kitadai, N. (2014) Thermodynamic prediction of glycine polymerization as a function of temperature and pH consistent with experimentally obtained results. *J. Mol. Evol.* **78**, 171–187. doi:10.1007/s0023901496161

Shock, E. L., Helgeson, H. C. and Sverjensky, D. A. (1989) Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Standard partial molal properties of inorganic neutral species. *Geochim. Cosmochim. Acta* **53**, 2157–2183. doi:10.1016/00167037(89)903414

Stefánsson, A. (2001) Dissolution of primary minerals of basalt in natural waters. I. Calculation of mineral solubilities from 0°C to 350°C. *Chem. Geol.* **172**, 225–250. doi:10.1016/S0009-2541(00)002631

Sverjensky, D. A., Shock, E. L., and Helgeson, H. C. (1997) Prediction of the thermodynamic properties of aqueous metal complexes to 1000 °C and 5 kbar. *Geochim. Cosmochim. Acta* **61**, 1359–1412. doi:10.1016/S00167037(97)000094

See Also

`thermo` (description of OBIGT), `mod.buffer` (modify buffer definitions), `logB.to.OBIGT` (fit thermodynamic parameters to formation constants)

Examples

```
## Modify an existing species (not real properties)
ialanine <- mod.OBIGT("alanine", state = "cr", G = 0, H = 0, S = 0)
# We have made the values of G, H, and S inconsistent
# with the elemental composition of alanine, so the following
# now produces a message about that
info(ialanine)

## Add an aqueous species (default) with Gibbs energy given in J/mol
## (the same as the default) and today's date
date <- as.character(Sys.Date())
iCl2O <- mod.OBIGT("Cl2O", date = date, E_units = "J", G = 87738)
info(iCl2O)

## Add a solid species with a name that is different from the formula
mod.OBIGT("lorem-ipsum", formula = "C123", state = "cr", G = -12345678)
# Retrieve the data for this species using either name or formula
info(info("lorem-ipsum"))
info(info("C123"))
# Reset database
OBIGT()

## Using add.OBIGT():
# Compare stepwise stability constants of cadmium chloride complexes
# using data from Sverjensky et al., 1997 and Bazarkina et al., 2010
Cdspecies <- c("Cd+2", "CdCl+", "CdCl2", "CdCl3-", "CdCl4-2")
P <- c(1, seq(25, 1000, 25))
SSH97 <- lapply(1:4, function(i) {
  subcrt(c(Cdspecies[i], "Cl-", Cdspecies[i+1]),
    c(-1, -1, 1), T=25, P=P)$out$logK
```

```
})
file <- system.file("extdata/adds/BZA10.csv", package="CHNOSZ")
add.OBIGT(file)
BZA10 <- lapply(1:4, function(i) {
  subcrt(c(Cdspecies[i], "Cl-", Cdspecies[i+1]),
    c(-1, -1, 1), T=25, P=P)$out$logK
})
matplot(P, do.call(cbind, SSH97), type="l")
matplot(P, do.call(cbind, BZA10), type="l", add=TRUE, lwd=2)
legend("topleft", legend=c("", "", "Sverjensky et al., 1997",
  "Bazarkina et al., 2010"), lwd=c(0, 0, 1, 2), bty="n")
# Make reaction labels
y <- c(1.8, 0.2, -0.5, -1)
invisible(lapply(1:4, function(i) {
  text(800, y[i], describe.reaction(subcrt(c(Cdspecies[i], "Cl-",
    Cdspecies[i+1]), c(-1, -1, 1), T=25, P=1)$reaction))
}))
# Restore default database
OBIGT()

# Another use of add.OBIGT()
# Compare Delta G of AABB = UPBB + H2O
# (Figure 9 of Kitadai, 2014)
# Default database has values from Kitadai, 2014
Kit14 <- subcrt(c("[AABB]", "[UPBB]", "H2O"), c(-1, 1, 1), T = seq(0, 300, 10))
# Load superseded parameters for [UPBB] from Dick et al., 2006
mod.OBIGT("[UPBB]", G = -21436, H = -45220, S = 1.62)
DLH06 <- subcrt(c("[AABB]", "[UPBB]", "H2O"), c(-1, 1, 1), T = seq(0, 300, 10))
xlab <- axis.label("T"); ylab <- axis.label("DG", prefix="k")
plot(Kit14$out$T, Kit14$out$G/1000, type = "l", ylim = c(10, 35),
     xlab = xlab, ylab = ylab)
lines(DLH06$out$T, DLH06$out$G/1000, lty = 2)
legend("topleft", c("Dick et al., 2006", "Kitadai, 2014"), lty = c(2, 1))
title(main = "AABB = UPBB + H2O; after Figure 9 of Kitadai, 2014")
# Restore default database
OBIGT()

# Another use of add.OBIGT(): calculate Delta G of
# H4SiO4 = SiO2 + 2H2O using various sources of data for SiO2.
# First, get H4SiO4 from Stefansson, 2001
add.OBIGT("AS04", "H4SiO4")
T <- seq(0, 350, 10)
s1 <- subcrt(c("H4SiO4", "SiO2", "H2O"), c(-1, 1, 2), T = T)
# Now, get SiO2 from Apps and Spycher, 2004
add.OBIGT("AS04", "SiO2")
s2 <- subcrt(c("H4SiO4", "SiO2", "H2O"), c(-1, 1, 2), T = T)
# Plot logK from the first and second calculations
plot(T, s1$out$G, type = "l", xlab = axis.label("T"),
  ylab = axis.label("DG"), ylim = c(-500, 2500))
lines(T, s2$out$G, lty = 2)
# Add title and legend
title(main = describe.reaction(s1$reaction))
stxt <- lapply(c("H4SiO4", "SiO2", "SiO2"), expr.species)
```

```
legend("top", c("Shock et al., 1989", "Apps and Spycher, 2004"),
  title = as.expression(expr.species("SiO2")), lty = c(1, 2))
legend("topright", "Stef\u00e1nsson, 2001",
  title = as.expression(expr.species("H4SiO4")))
abline(h = 0, lty = 3, col = 8)
# Take-home message: SiO2 from Ste01 is compatible with H4SiO4 from Ste01
# at low T, but SiO2 from Shock et al., 1989 (the default in OBIGT) isn't
OBIGT()
```

`add.protein` *Amino Acid Compositions of Proteins*

Description

Functions to get amino acid compositions and add them to protein list for use by other functions.

Usage

```
add.protein(aa, as.residue = FALSE)
seq2aa(sequence, protein = NA)
aasum(aa, abundance = 1, average = FALSE, protein = NULL, organism = NULL)
```

Arguments

`aa`	data frame, amino acid composition in the format of `thermo()$protein`
`as.residue`	logical, normalize by protein length?
`sequence`	character, protein sequence
`protein`	character, name of protein; numeric, indices of proteins (rownumbers of `thermo()$protein`)
`abundance`	numeric, abundances of proteins
`average`	logical, return the weighted average of amino acid counts?
`organism`	character, name of organism

Details

A '`protein`' in CHNOSZ is defined by its identifying information and the amino acid composition, stored in `thermo$protein`. The names of proteins in CHNOSZ are distinguished from those of other chemical species by having an underscore character ("_") that separates two identifiers, referred to as the `protein` and `organism`. An example is '`LYSC_CHICK`'. The purpose of the functions described here is to identify proteins and work with their amino acid compositions. From the amino acid compositions, the thermodynamic properties of the proteins can be estimated by group additivity.

`seq2aa` returns a data frame of amino acid composition for the provided `sequence`, in the format of `thermo()$protein`. In this function, the value of the `protein` argument is put into the `protein` column of the result. If there is an underscore (e.g. '`LYSC_CHICK`'), it is used to split the text, and the two parts are put into the `protein` and `organism` columns.

Given amino acid compositions returned by `seq2aa`, `add.protein` adds them to `thermo()$protein` for use by other functions in CHNOSZ. The amino acid compositions of proteins in `aa` with the same name as one in `thermo()$protein` are replaced. Set `as.residue` to TRUE to normalize by protein length; each input amino acid composition is divided by the corresponding number of residues, with the result that the sum of amino acid frequencies for each protein is 1.

`aasum` returns a data frame representing the sum of amino acid compositions in the rows of the input `aa` data frame. The amino acid compositions are multiplied by the indicated `abundance`; that argument is recycled to match the number of rows of `aa`. If `average` is TRUE the final sum is divided by the number of input compositions. The name used in the output is taken from the first row of `aa` or from `protein` and `organism` if they are specified.

Value

For `seq2aa`, a data frame of amino acid composition and identifying information for proteins. For `add.protein`, the rownumbers of `thermo()$protein` that are added and/or replaced. For `aasum`, a one-row data frame of amino acid composition and identifying information.

See Also

`read.fasta` for another way of getting amino acid compositions that can be used with `add.protein`.

`pinfo` for protein-level functions (length, chemical formulas, reaction coefficients of basis species).

Examples

```
# Get the amino acid composition of a protein sequence
# (Human Gastric juice peptide 1)
aa <- seq2aa("LAAGKVEDSD", "GAJU_HUMAN")
# Add the protein to CHNOSZ
ip <- add.protein(aa)
# Calculate the protein length and chemical formula
protein.length(ip) # 10
as.chemical.formula(protein.formula(ip)) # "C41H69N11O18"

# Calculate a formula without using add.protein
aa <- seq2aa("ANLSG", "pentapeptide_test")
as.chemical.formula(protein.formula(aa))

# Sum the amino acid compositions of several poliovirus protein subunits
file <- system.file("extdata/protein/POLG.csv", package = "CHNOSZ")
aa <- read.csv(file, as.is = TRUE)
aasum(aa, protein = "POLG_sum")
```

---

affinity *Chemical Affinities of Formation Reactions*

Description

Calculate the chemical affinities of formation reactions of species.

Usage

```
affinity(..., property = NULL, sout = NULL, exceed.Ttr = FALSE,
  exceed.rhomin = FALSE, return.buffer = FALSE, return.sout = FALSE,
  balance = "PBB", iprotein = NULL, loga.protein = 0, transect = NULL)
```

Arguments

`...`	numeric, zero or more named arguments, used to identify the variables of interest in the calculations. For argument recall, pass the output from a previous calculation of `affinity` as an unnamed first argument.
`property`	character, the property to be calculated. Default is 'A', for chemical affinity of formation reactions of species of interest
`sout`	list, output from `subcrt`
`exceed.Ttr`	logical, allow `subcrt` to compute properties for phases beyond their transition temperature?
`exceed.rhomin`	logical, allow `subcrt` to compute properties of species in the HKF model below 0.35 g cm^{-3}?
`return.buffer`	logical. If `TRUE`, and a `buffer` has been associated with one or more basis species in the system, return the values of the activities of the basis species calculated using the buffer. Default is `FALSE`.
`return.sout`	logical, return only the values calculated with `subcrt`?
`balance`	character. This argument is used to identify a conserved basis species (or 'PBB') in a chemical activity buffer. Default is 'PBB'.
`iprotein`	numeric, indices of proteins in `thermo$protein` for which to calculate properties
`loga.protein`	numeric, logarithms of activities of proteins identified in `iprotein`
`transect`	logical, force a transect calculation, even for three or fewer values of the variables?

Details

`affinity` calculates the chemical affinities of reactions to form the species of interest from the basis species. The equation used to calculate chemical affinity (A), written for base-10 (decimal) logarithms, is $A/(2.303RT)$=log(K/Q), where K is the equilibrium constant of the reaction, Q is the activity product of the species in the reaction, and 2.303 is the conversion factor from natural to decimal logarithms. The calculation of chemical affinities relies on the current definitions of the `basis` species and `species` of interest. Calculations are possible at single values of temperature, pressure, ionic strength and chemical activities of the basis species, or as a function of one or more of these variables.

The argument `property` can be changed to calculate other thermodynamic properties of formation reactions. Valid properties are 'A' or NULL for chemical affinity, 'logK' or 'logQ' for logarithm of equilibrium constant and reaction activity product, or any of the properties available in `subcrt` except for 'rho'. The properties returned are those of the formation reactions of the

species of interest from the basis species. It is also possible to calculate the properties of the species of interest themselves (not their formation reactions) by setting the property to 'G.species', 'Cp.species', etc. Except for 'A', the properties of proteins or their reactions calculated in this manner are restricted to nonionized proteins.

Zero, one, or more leading arguments to the function identify which of the chemical activities of basis species, temperature, pressure and/or ionic strength to vary. The names of each of these arguments may be the formula of any of the basis species of the system, or 'T', 'P', 'pe', 'pH', 'Eh', or 'IS' (but names may not be repeated). The names of charged basis species such as 'K+' and 'SO4-2' should be quoted when used as arguments. The value of each argument is of the form c(min, max) or c(min, max, res) where min and max refer to the minimimum and maximum values of variable identified by the name of the argument, and res is the resolution, or number of points along which to do the calculations; res is assigned a default value of 256 if it is missing. For any arguments that refer to basis species, the numerical values are the logarithms of activity (or fugacity for gases) of that basis species.

If 'T', 'P', and/or 'IS' are not among the vars, their constant values can be supplied in T, P, or IS (in mol kg^{-1}). The units of 'T' and 'P' are those set by T.units and P.units (on program start-up these are °C and bar, respectively). sout, if provided, replaces the call to subcrt, which can greatly speed up the calculations if this intermediate result is stored by other functions. exceed.Ttr is passed to subcrt so that the properties of mineral phases beyond their transition temperatures can optionally be calculated.

If one or more buffers are assigned to the definition of basis species, the logarithms of activities of these basis species are taken from the buffer (see buffer).

The iprotein and loga.protein arguments can be used to compute the chemical affinities of formation reactions of proteins that are not in the current species definition. iprotein contains the indices (rownumbers) of desired proteins in thermo$protein. This uses some optimizations to calculate the properties of many proteins in a fraction of the time it would take to calculate them individually.

When the length(s) of the variables is(are) greater than 3, the function enters the 'transect' mode of operation. In this mode of operation, instead of performing the calculations on an n-dimensional grid, the affinities are calculated on a transect of changing T, P, and/or chemical activity of basis species.

Argument recall is invoked by passing a previous result of affinity as the first argument. The function then calls itself using the settings from the previous calculation, with additions or modifications indicated by the remaining arguments in the current function call.

Value

A list, elements of which are fun the name of the function ('affinity'), args all of the arguments except for 'sout' (these are used for argument recall), sout output from subcrt, property name of the calculated property ('A' for chemical affinity), basis and species definition of basis species and species of interest in effect at runtime, T and P temperature and pressure, in the system units of Kelvin and bar, set to numeric() (length=0) if either one is a variable, vars the names of the variables, vals the values of the variables (a list, one element for each variable), values the result of the calculation (a list, one element for each species, with names taken from the species index in thermo$OBIGT). The elements of the lists in vals and values are arrays of n dimensions, where n is the number of variables. The values of chemi-

cal affinity of formation reactions of the species are returned in dimensionless units (for use with decimal logarithms, i.e., $A/2.303RT$).

Names other than 'T' or 'P' in vars generally refer to basis species, and the corresponding vals are the logarithms of activity or fugacity. However, if one or more of pe, Eh or pH is among the variables of interest, vals holds the values of the those variables as indicated.

References

Helgeson, H. C., Richard, L, McKenzie, W. F., Norton, D. L. and Schmitt, A. (2009) A chemical and thermodynamic model of oil generation in hydrocarbon source rocks. *Geochim. Cosmochim. Acta* **73**, 594–695. doi:10.1016/j.gca.2008.03.004

See Also

ionize.aa, activated if proteins are among the species of interest, 'H+' is in the basis and thermo()optionize.aa is TRUE. equilibrate for using the results of affinity to calculate equilibrium activities of species, and diagram to plot the results. demo("saturation") for an example using the argument recall feature.

Examples

```
## Set up a system and calculate
## chemical affinities of formation reactions
basis(c("SiO2", "MgO", "H2O", "O2"), c(-5, -5, 0, 999))
species(c("quartz","enstatite","forsterite"))
# Chemical affinities (A/2.303RT) at 25 deg C and 1 bar
affinity()
# At higher temperature and pressure
affinity(T = 500, P = 2000)
# At 25 temperatures and pressures,
# some are in the low-density region so we suppress warnings
suppressWarnings(affinity(T = c(500, 1000, 5), P = c(1000, 5000, 5)))
# Equilibrium constants of formation reactions
affinity(property = "logK")
# Standard molal Gibbs energies of species,
# in units set by E.units() (default: J/mol)
affinity(property = "G.species")
# Standard molal Gibbs energies of reactions
affinity(property = "G")
# A T,P-transect
# (fluid pressure from Helgeson et al., 2009 Fig. 7)
affinity(T = c(25, 110, 115, 215), P = c(11, 335, 500, 1450))
```

---

basis *Define Basis Species*

Description

Define the basis species of a chemical system.

Usage

```
basis(species = NULL, state = NULL, logact = NULL,
  delete = FALSE, add = FALSE)
```

Arguments

species	character, names or formulas of species, or numeric, indices of species
state	character, physical states or names of buffers
logact	numeric, logarithms of activities or fugacities
delete	logical, delete the current basis definition?
add	logical, add species to the current basis definition?

Details

The basis species represent the possible range of chemical compositions for all the species of interest. As used here, a set of basis species is valid only if it satisifes two conditions: 1) the number of basis species is the same as the number of chemical elements (including charge) in those species and 2) the square matrix representing the elemental stoichiometries of the basis species has a real inverse.

To create a basis definition, call `basis` with the names or formulas of the basis species in the `species` argument, or all numeric values as species indices (rownumbers in `thermo()$OBIGT`). The special names 'pH', 'pe' and 'Eh' can also be used; they get translated into the names of the proton ('H+') and electron ('e-') as appropriate. If desired, include the `state` for the named species and the logarithms of activity (fugacity for gases) in `logact`. The latter defaults to zero (unit activity) if not specified.

To modify an existing basis definition, the physical states or logarithms of activities of species can be changed by calling `basis` with a `species` argument that has the formulas (not names) or indices of species in the existing basis. If either of the second or third arguments to `basis` is of type character, it refers to the physical state (if present in `thermo()$OBIGT$state`) or a chemical activity `buffer` (if present in `thermo()$buffers$name`). If either of these arguments is numeric it specifies the logarithms of activities (or fugacities for gases) of the basis species. In case 'pH', 'pe' or 'Eh' is named, the logarithm of activity of the basis species is converted from these values. For example, a value of 7 for pH is stored as a logarithm of activity of -7.

If `add` is TRUE, then the function attempts to *add* the indicated `species` to the basis definition. This only works if the enlarged set of species is a valid basis set as described above. If the formed `species` are currently defined, their formation reactions are modified accordingly (with zeroes for the newly added basis species).

If `add` is FALSE, and if `basis` is called with NULL values of both `state` and `logact`, the new set of species, if they are a valid basis set, completely replaces any existing basis definition. When this occurs, any existing species definition (created by the `species` function) is deleted. Call `basis` with `delete` set to TRUE or `species` set to '""' to clear the basis definition and that of the `species`, if present.

If the value of `basis` is one of the keywords in the following table, the corresponding set of basis species is loaded, and their activities are given preset values. The basis species identified by these keywords are aqueous except for H_2O (liq), O_2 (gas) and Fe_2O_3 (hematite).

`CHNOS`	CO_2, H_2O, NH_3, H_2S, O_2
`CHNOS+`	CO_2, H_2O, NH_3, H_2S, O_2, H^+
`CHNOSe`	CO_2, H_2O, NH_3, H_2S, e^-, H^+
`CHNOPS+`	CO_2, H_2O, NH_3, H_3PO_4, H_2S, O_2, H^+
`CHNOPSe`	CO_2, H_2O, NH_3, H_3PO_4, H_2S, e^-, H^+
`MgCHNOPS+`	Mg^{+2}, CO_2, H_2O, NH_3, H_3PO_4, H_2S, O_2, H^+
`MgCHNOPSe`	Mg^{+2}, CO_2, H_2O, NH_3, H_3PO_4, H_2S, e^-, H^+
`FeCHNOS`	Fe_2O_3, CO_2, H_2O, NH_3, H_2S, O_2
`FeCHNOS+`	Fe_2O_3, CO_2, H_2O, NH_3, H_2S, O_2, H^+
`QEC4`	cysteine, glutamic acid, glutamine, H_2O, O_2
`QEC`	cysteine, glutamic acid, glutamine, H_2O, O_2
`QEC+`	cysteine, glutamic acid, glutamine, H_2O, O_2, H^+
`QCa`	glutamine, cysteine, acetic acid, H_2O, O_2
`QCa+`	glutamine, cysteine, acetic acid, H_2O, O_2, H^+

The logarithms of activities of amino acids in the '`QEC4`' basis are -4 (i.e., basis II in Dick, 2016); those in '`QEC`' and '`QEC+`' are set to approximate concentrations in human plasma (see Dick, 2017).

Value

Returns the value of `thermo()$basis` after any modifications; or, if `delete` is TRUE, its value before deletion (invisibly).

References

Dick, J. M. (2016) Proteomic indicators of oxidation and hydration state in colorectal cancer. *PeerJ* **4**:e2238. doi:10.7717/peerj.2238

Dick, J. M. (2017) Chemical composition and the potential for proteomic transformation in cancer, hypoxia, and hyperosmotic stress. *PeerJ* **5**:e3421 doi:10.7717/peerj.3421

See Also

`info` to query the thermodynamic database in order to find what species are available. `makeup` is used by `basis` to generate the stoichiometric matrix from chemical formulas. `swap.basis` is used to change the chemical compounds (species formulas) used in the basis definition while keeping the chemical potentials of the elements unaltered. `species` for setting up the formation reactions from basis species.

Examples

```
## Define basis species
# with one, two or three elements
basis("O2")
basis(c("H2O", "O2"))
basis(c("H2O", "O2", "H+"))
## Clear the basis species
basis("")
```

```
## Not run:
## Marked dontrun because they produce errors
# Fewer species than elements
basis(c("H2O", "H+"))
# More species than elements
basis(c("H2O", "O2", "H2", "H+"))
# Non-independent species
basis(c("CO2", "H2O", "HCl", "Cl-", "H+"))
## End(Not run)

## Specify activities and states
basis(c("H2O", "O2", "CO2"), c(-2, -78, -3), c("liq", "aq", "aq"))
# Change logarithms of activities/fugacities
basis(c("H2O", "O2"), c(0, -80))
# Change state of CO2
basis("CO2", "gas")
```

`Berman` *Thermodynamic Properties of Minerals*

Description

Calculate thermodynamic properties of minerals using the equations of Berman (1988).

Usage

```
Berman(name, T = 298.15, P = 1, check.G = FALSE,
       calc.transition = TRUE, calc.disorder = TRUE)
```

Arguments

`name`	character, name of mineral
`T`	numeric, temperature(s) at which to calculate properties (K)
`P`	numeric, pressure(s) at which to calculate properties (bar)
`check.G`	logical, check consistency of G, H, and S?
`calc.transition`	logical, include calculation of polymorphic transition properties?
`calc.disorder`	logical, include calculation of disordering properties?

Details

This function calculates the thermodynamic properties of minerals at high P and T using equations given by Berman (1988). These minerals should be listed in `thermo()$OBIGT` with the state '`cr`' and chemical formula, and optionally an abbreviation and references, but all other properties set to NA.

The standard state thermodynamic properties and parameters for the calculations are stored in data files under `extdata/Berman`, or can be read from a user-created file specified by `thermo()$opt$Berman`.

The equation used for heat capacity is C_P = k0 + k1*$T^{-0.5}$ + k2*T^{-2} + k3*T^{-3} + k4*T^{-1} + k5*T + k6*T^2. This is an extended form Eq. 4 of Berman (1988) as used in the winTWQ program (Berman, 2007). The equation used for volume is $V(P, T)$ / V(1 bar, 298.15 K) = 1 + v1 * (T - 298.15) + v2 * (T - 298.15)^2 + v3 * (P - 1) + v4 * (P - 1)^2 (Berman, 1988, Eq. 5, with terms reordered to follow winTWQ format). The equations used for lambda transitions follow Eqs. 8-14 of Berman (1988). The equation used for the disorder contribution between Tmin and Tmax is C_P[dis] = d0 + d1*$T^{-0.5}$ + d2*T^{-2} + d3*T + d4*T^2 (Berman, 1988, Eq. 15). The parameters correspond to Tables 2 (`GfPrTr`, `HfPrTr`, `SPrTr`, `VPrTr`), 3a (`k0` to `k3`), 4 (`v1` to `v4`), 3b (transition parameters: `Tlambda` to `dTH`), and 5 (disorder parameters: `Tmax`, `Tmin`, `d1` to `d4` and `Vad`) of Berman (1988). Following the winTWQ data format, multipliers are applied to the volume parameters only (see below). Note that `VPrTr` is tabulated in J bar^{-1} mol^{-1}, which is equal to 10 cm^3 mol^{-1}.

A value for `GfPrTr` is not required and is only used for optional checks (see below). Numeric values (possibly 0) should be assigned for all of `HfPrTr`, `SPrTr`, `VPrTr`, `k0` to `k6` and `v1` to `v4`. Missing (or NA) values are permitted for the transition and disorder parameters, for minerals where they are not used. The data files have the following 30 columns:

`name`	mineral name (must match an entry with a formula but NA properties in `thermo()$OBIGT`)
`GfPrTr`	standard Gibbs energy at 298.15 K and 1 bar (J mol^{-1}) (Benson-Helgeson convention)
`HfPrTr`	standard enthalpy at 298.15 K and 1 bar (J mol^{-1})
`SPrTr`	standard entropy at 298.15 K and 1 bar (J mol^{-1} K^{-1})
`VPrTr`	standard volume at 298.15 K and 1 bar (J bar^{-1}) [1 J bar^{-1} = 10 cm^3]
`k0` ... `k6`	k0 (J mol^{-1} K^{-1}) to k6
`v1`	v1 (K^{-1}) * 10^5
`v2`	v2 (K^{-2}) * 10^5
`v3`	v3 (bar^{-1}) * 10^5
`v4`	v4 (bar^{-2}) * 10^8
`Tlambda`	T_λ (K)
`Tref`	T_{ref} (K)
`dTdP`	dT / dP (K bar^{-1})
`l1`	l1 ((J/mol)$^{0.5}$ K^{-1})
`l2`	l2 ((J/mol)$^{0.5}$ K^{-2})
`DtH`	$\Delta_t H$ (J mol^{-1})
`Tmax`	temperature at which phase is fully disordered (T_D in Berman, 1988) (K)
`Tmin`	reference temperature for onset of disordering (t in Berman, 1988) (K)
`d0` ... `d4`	d0 (J mol^{-1} K^{-1}) to d4
`Vad`	constant that scales the disordering enthalpy to volume of disorder (d_5 in Berman, 1988)

The function outputs apparent Gibbs energies according to the Benson-Helgeson convention (ΔG = ΔH - $T\Delta S$) using the entropies of the elements in the chemical formula of the mineral to calculate ΔS (cf. Anderson, 2005). If `check.G` is TRUE, the tabulated value of `GfTrPr` (Benson-Helgeson) is compared with that calculated from `HfPrTr` - 298.15*`DSPrTr` (`DSPrTr` is the difference between the entropies of the elements in the formula and `SPrTr` in the table). A warning is produced if the absolute value of the difference between tabulated and calculated `GfTrPr` is greater than 1000 J/mol.

If the function is called with missing `name`, the parameters for all available minerals are returned.

Value

A data frame with `T` (K), `P` (bar), `G`, `H`, `S`, and `Cp` (energetic units in Joules), and `V` (cm^3 mol^{-1}).

References

Anderson, G. M. (2005) *Thermodynamics of Natural Systems*, 2nd ed., Cambridge University Press, 648 p. https://www.worldcat.org/oclc/474880901

Berman, R. G. (1988) Internally-consistent thermodynamic data for minerals in the system Na_2O-K_2O-CaO-MgO-FeO-Fe_2O_3-Al_2O_3-SiO_2-TiO_2-H_2O-CO_2. *J. Petrol.* **29**, 445-522. doi:10.1093/petrology/29.2.445

Berman, R. G. and Aranovich, L. Ya. (1996) Optimized standard state and solution properties of minerals. I. Model calibration for olivine, orthopyroxene, cordierite, garnet, and ilmenite in the system FeO-MgO-CaO-Al_2O_3-TiO_2-SiO_2. *Contrib. Mineral. Petrol.* **126**, 1-24. doi:10.1007/s004100050233

Berman, R. G. (2007) winTWQ (version 2.3): A software package for performing internally-consistent thermobarometric calculations. *Open File* **5462**, Geological Survey of Canada, 41 p. doi:10.4095/223425

Helgeson, H. C., Delany, J. M., Nesbitt, H. W. and Bird, D. K. (1978) Summary and critique of the thermodynamic properties of rock-forming minerals. *Am. J. Sci.* **278-A**, 1–229. https://www.worldcat.org/oclc/13594862

Examples

```
# Other than the formula, the parameters aren't stored in
# thermo()$OBIGT, so this shows NAs
info(info("quartz", "cr"))
# Properties of alpha-quartz (aQz) at 298.15 K and 1 bar
Berman("quartz")
# Gibbs energies of aQz and coesite at higher T and P
T <- seq(200, 1300, 100)
P <- seq(22870, 31900, length.out = length(T))
G_aQz <- Berman("quartz", T = T, P = P)$G
G_Cs <- Berman("coesite", T = T, P = P)$G
# That is close to the univariant curve (Ber88 Fig. 4),
# so the difference in G is close to 0
DGrxn <- G_Cs - G_aQz
all(abs(DGrxn) < 100)  # TRUE

# Make a P-T diagram for SiO2 minerals (Ber88 Fig. 4)
basis(c("SiO2", "O2"), c("cr", "gas"))
species(c("quartz", "quartz,beta", "coesite"), "cr")
a <- affinity(T = c(200, 1700, 200), P = c(0, 50000, 200))
diagram(a)

## Getting data from a user-supplied file
## Ol-Opx exchange equilibrium, after Berman and Aranovich, 1996
species <- c("fayalite", "enstatite", "ferrosilite", "forsterite")
```

```
coeffs <- c(-1, -2, 2, 1)
T <- seq(600, 1500, 50)
Gex_Ber88 <- subcrt(species, coeffs, T = T, P = 1)$out$G
# Add data from BA96
datadir <- system.file("extdata/Berman/testing", package = "CHNOSZ")
add.OBIGT(file.path(datadir, "BA96_OBIGT.csv"))
thermo("opt$Berman" = file.path(datadir, "BA96_Berman.csv"))
Gex_BA96 <- subcrt(species, coeffs, T = seq(600, 1500, 50), P = 1)$out$G
# Ber88 is lower than BA96 at low T
(Gex_BA96 - Gex_Ber88)[1] > 0  # TRUE
# The curves cross at about 725 deg C (BA96 Fig. 8)
# (actually, in our calculation they cross closer to 800 deg C)
T[which.min(abs(Gex_BA96 - Gex_Ber88))]  # 800
# Reset the database (thermo()$OBIGT, and thermo()$opt$Berman)
reset()
```

`buffer` *Calculating Buffered Chemical Activities*

Description

Calculate values of activity or fugacity of basis species buffered by an assemblage of one or more species.

Usage

```
mod.buffer(name, species = NULL, state = "cr", logact = 0)
```

Arguments

`name`	character, name of buffer to add to or find in `thermo()$buffer`.
`species`	character, names or formulas of species in a buffer.
`state`	character, physical states of species in buffer.
`logact`	numeric, logarithms of activities of species in buffer.

Details

A buffer is treated here as assemblage of one or more species whose presence constrains values of the chemical activity (or fugacity) of one or more basis species. To perform calculations for buffers use `basis` to associate the name of the buffer with one or more basis species. After this, calls to `affinity` will invoke the required calculations. The calculated values of the buffered activites can be retrieved by setting `return.buffer` to `TRUE` (in `affinity`). The maximum number of buffered chemical activities possible for any buffer is equal to the number of species in the buffer; however, the user may then elect to work with the values for only one or some of the basis species calculated with the buffer.

The identification of a conserved basis species (or other reaction balancing rule) is required in calculations for buffers of more than one species. For example, in the pyrite-pyrrhotite-magnetite

buffer (FeS_2-FeS-Fe_3O_4) a basis species common to each species is one representing Fe. Therefore, when writing reactions between the species in this buffer Fe is conserved while H_2S and O2 are the variables of interest. The calculation for buffers attempts to determine which of the available basis species qualifies as a conserved quantity. This can be overriden with `balance`. The default value of `balance` is '`PBB`', which instructs the function to use the protein backbone group as the conserved quantity in buffers consisting of proteins, but has no overriding effect on the computations for buffers without proteins.

To view the available buffers, print the `thermo()$buffer` object. Buffer definitions can be added to this dataframe with `mod.buffer`. The defaults for `state` and `logact` are intended for mineral buffers. If `name` identifies an already defined buffer, this function modifies the logarithms of activities or states of species in that buffer, optionally restricted to only those species given in `species`.

It is possible to assign different buffers to different basis species, in which case the order of their calculation depends on their order in `thermo()$buffers`. This function is compatible with systems of proteins, but note that for buffers *made* of proteins the buffer calculations presently use whole protein formulas (instead of residue equivalents) and consider nonionized proteins only.

References

Garrels, R. M. (1960) *Mineral Equilibria*. Harper & Brothers, New York, 254 p. https://www.worldcat.org/oclc/552690

See Also

`diagram` with `type` set to the name of a basis species solves for the activity of the basis species.

Examples

```
## List the buffers
thermo()$buffer
# Another way to do it, for a specific buffer
print(mod.buffer("PPM"))

## Buffer made of one species
# Calculate the activity of CO2 in equilibrium with
# (a buffer made of) acetic acid at a given activity
basis("CHNOS")
basis("CO2", "AC")
# What activity of acetic acid are we using?
print(mod.buffer("AC"))
# Return the activity of CO2
affinity(return.buffer = TRUE)$CO2  # -7.057521
# As a function of oxygen fugacity
affinity(O2 = c(-85, -70, 4), return.buffer = TRUE)
# As a function of logfO2 and temperature
affinity(O2 = c(-85, -70, 4), T = c(25, 100, 4), return.buffer = TRUE)
# Change the activity of species in the buffer
mod.buffer("AC", logact = -10)
affinity(O2 = c(-85,-70,4), T = c(25, 100, 4), return.buffer = TRUE)
```

```
## Buffer made of three species
## Pyrite-Pyrrhotite-Magnetite (PPM)
# Specify basis species and initial activities
basis(c("FeS2", "H2S", "O2", "H2O"), c(0, -10, -50, 0))
# Note that the affinity of formation of pyrite,
# which corresponds to FeS2 in the basis, is zero
species(c("pyrite", "pyrrhotite", "magnetite"))
affinity(T = c(200, 400, 11), P = 2000)$values
# Setup H2S and O2 to be buffered by PPM
basis(c("H2S", "O2"), c("PPM", "PPM"))
# Inspect values of H2S activity and O2 fugacity
affinity(T = c(200, 400, 11), P = 2000, return.buffer = TRUE, exceed.Ttr = TRUE)
# Calculate affinities of formation reactions of species in the buffer
a <- affinity(T = c(200, 400, 11), P = 2000, exceed.Ttr = TRUE)$values
# The affinities for species in the buffer are all equal to zero
all.equal(as.numeric(a[[1]]), rep(0, 11))  # TRUE
all.equal(as.numeric(a[[2]]), rep(0, 11))  # TRUE
all.equal(as.numeric(a[[3]]), rep(0, 11))  # TRUE

## Buffer made of one species: show values of logfO2 on an
## Eh-pH diagram; after Garrels, 1960, Figure 6
basis("CHNOSe")
# Here we will buffer the activity of the electron by O2
mod.buffer("O2", "O2", "gas", 999)
basis("e-", "O2")
# Start our plot, then loop over values of logfO2
thermo.plot.new(xlim = c(0, 14), ylim = c(-0.8, 1.2),
  xlab = "pH",ylab = axis.label("Eh"))
# The upper and lower lines correspond to the upper
# and lower stability limits of water
logfO2 <- c(0, -20, -40, -60, -83.1)
for(i in 1:5) {
  # Update the logarithm of fugacity (logact) of O2 in the buffer
  mod.buffer("O2", "O2", "gas", logfO2[i])
  # Get the values of the logarithm of activity of the electron
  a <- affinity(pH = c(0, 14, 15), return.buffer = TRUE)
  # Convert values of pe (-logact of the electron) to Eh
  Eh <- convert(-as.numeric(a$`e-`), "Eh")
  lines(seq(0, 14, length.out = 15), Eh)
  # Add some labels
  text(seq(0, 14, length.out = 15)[i*2+2], Eh[i*2+2],
    paste("logfO2 =", logfO2[i]))
}
title(main = paste("Relation between logfO2(g), Eh and pH at\n",
  "25 degC and 1 bar. After Garrels, 1960"))

## Buffer made of two species
# Conditions for metastable equilibrium among
# CO2 and acetic acid. note their starting activities:
print(mod.buffer("CO2-AC"))
basis("CHNOS")
basis("O2", "CO2-AC")
affinity(return.buffer = TRUE)  # logfO2 = -75.94248
```

```
basis("CO2", 123)  # what the buffer reactions are balanced on
affinity(return.buffer = TRUE)  # unchanged
# Consider more oxidizing conditions
mod.buffer("CO2-AC", logact = c(0, -10))
affinity(return.buffer = TRUE)
```

DEW *Deep Earth Water (DEW) Model*

Description

Calculate thermodynamic properties of water using the Deep Earth Water (DEW) model.

Usage

```
calculateDensity(pressure, temperature, error = 0.01)
calculateGibbsOfWater(pressure, temperature)
calculateEpsilon(density, temperature)
calculateQ(density, temperature)
```

Arguments

`pressure`	numeric, pressure (bar)
`temperature`	numeric, temperature (°C)
`error`	numeric, residual error for bisection calculation
`density`	numeric, density (g/cm^3)

Details

The Deep Earth Water (DEW) model, described by Sverjensky et al., 2014, extends the applicability of the revised HKF equations of state to 60 kbar. This implementation of DEW is based on the VBA macro code in the May, 2017 version of the DEW spreadsheet downloaded from http://www.dewcommunity.org/. The spreadsheet provides multiple options for some calculations; here the default equations for density of water (Zhang and Duan, 2005), dielectric constant (Sverjensky et al., 2014) and Gibbs energy of water (integral of volume, equation created by Brandon Harrison) are used.

Comments in the original code indicate that `calculateGibbsOfWater` is valid for $100 \leq T \leq 1000$ °C and $P \geq 1000$ bar. Likewise, the power function fit of the dielectric constant (epsilon) is valid for $100 \leq T \leq 1200$ °C and $P \geq 1000$ bar (Sverjensky et al., 2014).

Value

The calculated values of density, Gibbs energy, and the Q Born coefficient have units of g/cm^3, cal/mol, and bar^-1 (epsilon is dimensionless).

References

Sverjensky, D. A., Harrison, B. and Azzolini, D. (2014) Water in the deep Earth: The dielectric constant and the solubilities of quartz and corundum to 60 kb and 1,200 °C. *Geochim. Cosmochim. Acta* **129**, 125–145. doi:10.1016/j.gca.2013.12.019

Zhang, Z. and Duan, Z. (2005) Prediction of the *PVT* properties of water over wide range of temperatures and pressures from molecular dynamics simulation. *Phys. Earth Planet. Inter.* **149**, 335–354. doi:10.1016/j.pepi.2004.11.003

See Also

`water.DEW`; use `water("DEW")` to activate these equations for the main functions in CHNOSZ.

Examples

```
pressure <- c(1000, 60000)
temperature <- c(100, 1000)
calculateGibbsOfWater(pressure, temperature)
(density <- calculateDensity(pressure, temperature))
calculateEpsilon(density, temperature)
calculateQ(density, temperature)
```

---

diagram *Chemical Activity Diagrams*

Description

Plot equilibrium chemical activity (1-D speciation) or equal-activity (2-D predominance) diagrams as a function of chemical activities of basis species, temperature and/or pressure.

Usage

```
diagram(
  # species affinities or activities
  eout,
  # type of plot
  type = "auto", alpha = FALSE, normalize = FALSE,
  as.residue = FALSE, balance = NULL, groups = as.list(1:length(eout$values)),
  # figure size and sides for axis tick marks
  xrange = NULL, mar = NULL, yline = par("mgp")[1]+0.3, side = 1:4,
  # axis limits and labels
  ylog = TRUE, xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,
  # character sizes
  cex = par("cex"), cex.names = 1, cex.axis = par("cex"),
  # line styles
  lty = NULL, lty.cr = NULL, lty.aq = NULL, lwd = par("lwd"), dotted = NULL,
  spline.method = NULL, contour.method = "edge", levels = NULL,
  # colors
```

```
    col = par("col"), col.names = par("col"), fill = NULL,
    fill.NA = "gray80", limit.water = NULL,
    # field and line labels
    names = NULL, format.names = TRUE, bold = FALSE, italic = FALSE,
    font = par("font"), family = par("family"), adj = 0.5,
    dx = 0, dy = 0, srt = 0, min.area = 0,
    # title and legend
    main = NULL, legend.x = NA,
    # plotting controls
    add = FALSE, plot.it = TRUE, tplot = TRUE, ...)
  find.tp(x)
```

Arguments

`eout`	list, object returned by `affinity`, `equilibrate` or related functions
`type`	character, type of plot, or name of basis species whose activity to plot
`alpha`	logical or character ('`balance`'), for speciation diagrams, plot degree of formation instead of activities?
`normalize`	logical, divide chemical affinities by balance coefficients and rescale activities to whole formulas?
`as.residue`	logical, divide chemical affinities by balance coefficients (no rescaling)?
`balance`	character, balancing constraint; see `equilibrate`
`groups`	list of numeric, groups of species to consider as a single effective species
`xrange`	numeric, range of x-values between which predominance field boundaries are plotted
`mar`	numeric, margins of plot frame
`yline`	numeric, margin line on which to plot the y-axis name
`side`	numeric, which sides of plot to draw axes
`xlim`	numeric, limits of x-axis
`ylim`	numeric, limits of y-axis
`xlab`	character, label to use for x-axis
`ylab`	character, label to use for y-axis
`ylog`	logical, use a logarithmic y-axis (on 1D degree diagrams)?
`cex`	numeric, character expansion (scaling relative to current)
`cex.names`	numeric, character expansion factor to be used for names of species on plots
`cex.axis`	numeric, character expansion factor for names of axes
`lty`	numeric, line types to be used in plots
`lty.cr`	numeric, line types for cr-cr boundaries (between two minerals)
`lty.aq`	numeric, line types for aq-aq boundaries (between two aqueous species)
`lwd`	numeric, line width
`dotted`	numeric, how often to skip plotting points on predominance field boundaries (to gain the effect of dotted or dashed boundary lines)

spline.method	character, method used in splinefun
contour.method	character, labelling method used in contour (use NULL for no labels).
levels	numeric, levels at which to draw contour lines
col	character, color of activity lines (1D diagram) or predominance field boundaries (2D diagram)
col.names	character, colors for labels of species
fill	character, colors used to fill predominance fields
fill.NA	character, color for grid points with NA values
limit.water	NULL or logical, set NA values beyond water stability limits?
names	character, names of species for activity lines or predominance fields
format.names	logical, apply formatting to chemical formulas?
bold	logical, use bold formatting for names?
italic	logical, use italic formatting for names?
font	character, font type for names (has no effect if format.names is TRUE)
family	character, font family for names
adj	numeric, adjustment for line labels
dx	numeric, x offset for line or field labels
dy	numeric, y offset for line or field labels
srt	numeric, rotation for line labels
min.area	numeric, minimum area of fields that should be labeled, expressed as a fraction of the total plot area
main	character, a main title for the plot; NULL means to plot no title
legend.x	character, description of legend placement passed to legend
add	logical, add to current plot?
plot.it	logical, make a plot?
tplot	logical, set up plot with thermo.plot.new?
x	matrix, value of the predominant list element from diagram
...	additional arguments passed to plot or barplot

Details

This function displays diagrams representing either chemical affinities or equilibrium chemical activities of species. The first argument is the output from affinity, rank.affinity, equilibrate, or solubility. 0-D diagrams, at a single point, are shown as barplots. 1-D diagrams, for a single variable on the x-axis, are plotted as lines. 2-D diagrams, for two variables, are plotted as predominance fields. The allowed variables are any that affinity or the other functions accepts: temperature, pressure, or the chemical activities of the basis species.

A new plot is started unless add is TRUE. If plot.it is FALSE, no plot will be generated but all the intermediate computations will be performed and the results returned.

Line or field labels use the names of the species as provided in `eout`; formatting is applied to chemical formulas (see `expr.species`) unless `format.names` is FALSE. Set `names` to TRUE or NULL to plot the names, or FALSE, NA, or `""` to prevent plotting the names, or a character argument to replace the default species names. Alternatively, supply a numeric value to `names` to indicate a subset of default names that should or shouldn't be plotted (positive and negative indices, respectively). Use `col.names` and `cex.names` to change the colors and size of the labels. Use `cex` and `cex.axis` to adjust the overall character expansion factors (see `par`) and those of the axis labels. The x- and y-axis labels are automatically generated unless they are supplied in `xlab` and `ylab`.

If `groups` is supplied, the activities of the species identified in each numeric element of this list are multiplied by the balance coefficients of the species, then summed together. The names of the list are used to label the lines or fields for the summed activities of the resulting groups.

Normalizing protein formulas by length gives "residue equivalents" (Dick and Shock, 2011) that are useful for equilibrium calculations with proteins. `normalize` and `as.residue` are only usable when `eout` is the output from `affinity`, and only one can be TRUE. If `normalize` is TRUE, formation reactions and their affinities are first divided by protein length, so equal activities of residue equivalents are considered; then, the residue activities are rescaled to whole proteins for making the plot. If `as.residue` is TRUE, no rescaling is performed, so the diagram represents activities of the residues, not the whole proteins.

`type` argument

This paragraph describes the effect of the `type` argument when the output from `affinity` is being used. For `type` set to '`auto`', and with 0 or 1 variables defined in `affinity`, the property computed by `affinity` for each species is plotted. This is usually the affinity of the formation reactions, but can be set to some other property (using the `property` argument of `affinity`), such as the equilibrium constant ('`logK`'). For two variables, equilibrium predominance (maximum affinity) fields are plotted. This "maximum affinity method" (Dick, 2019) uses balancing coefficients that are specified by the `balance` argument. If `type` is '`saturation`', the function plots the line for each species where the affinity of formation equals zero (see `demo("saturation")` for an example). If for a given species no saturation line is possible or the range of the diagram does not include the saturation line, the function prints a message instead. If `type` is the name of a basis species, then the equilibrium activity of the selected basis species in each of the formation reactions is plotted (see the CO_2-acetic acid example in `buffer`). In the case of 2-D diagrams, both of these options use `contour` to draw the lines, with the method specified in `contour.method`.

This paragraph describes the effect of the `type` argument when the output from `solubility` is being used. For one mineral or gas, if `type` set to '`auto`', the equilibrium activities of each aqueous species are plotted. If `type` is '`loga.balance`', the activity of the balancing basis species (i.e. total solubility) is plotted; this is represented by contours on a 2-D diagram. For two or more minerals or gases, if `type` set to '`auto`', the values of '`loga.balance`' (overall minimum solubility) are plotted. If `type` is '`loga.equil`', the solubilities of the individual minerals and gases are plotted. For examples that use these features, see `solubility` and various `demos`: '`DEW`', '`contour`', '`gold`', '`solubility`', '`sphalerite`'.

1-D diagrams

For 1-D diagrams, the default setting for the y-axis is a logarithmic scale (unless `alpha` is TRUE) with limits corresponding to the range of logarithms of activities (or 0,1 if `alpha` is TRUE); these

actions can be overridden by ylog and ylim. If legend.x is NA (the default), the lines are labeled with the names of the species near the maximum value. Otherwise, a legend is placed at the location identified by legend.x, or omitted if legend.x is NULL.

If alpha is TRUE, the fractional degrees of formation (ratios of activities to total activity) are plotted. Or, setting alpha to 'balance' allows the activities to be multiplied by the number of the balancing component; this is useful for making "percent carbon" diagrams where the species differ in carbon number. The line type and line width can be controlled with lty and lwd, respectively. To connect the points with splines instead of lines, set spline.method to one of the methods in splinefun.

2-D diagrams

On 2-D diagrams, the fields represent the species with the highest equilibrium activity. fill determines the color of the predominance fields, col that of the boundary lines. The default of NULL for fill uses a light blue, light tan, and darker tan color for fields with aqueous species, one solid, or two solids. These correspond to the web colors "aliceblue", "antiquewhite", and "burlywood" with some transparency added; see **multi-metal.html** for an example with two solids produced using mix. If all the species in the diagram have the same state, or if the fill argument is NA or a 0-length value, the predominance fields are transparent, i.e. no fill color is used. Otherwise, fill can be any colors, or the word 'rainbow', 'heat', 'terrain', 'topo', or 'cm', indicating a palette from **grDevices**. Starting with R version 3.6.0, fill can be the name of any available HCL color palette, matched in the same way as the palette argument of hcl.colors.

fill.NA gives the color for empty fields, i.e. points at which NA values are present for any species. This may occur when there are missing thermodynamic data or the temperature or pressure are not in the range of the equations of state. To make overlay diagrams easier to construct, the default for fill.NA is automatically changed to 'transparent' when add is TRUE.

If limit.water is TRUE, the diagram is clipped to the the water stability region on Eh-pH (and some other) diagrams. That is, predominance fields are shown only where water is stable, and fill.NA is used for areas where H_2O is not stable. The default of NULL for limit.water does not clip the main diagram but instead overlays it on the water stability fields. Change limit.water to FALSE to not show the water stability regions at all; this is automatically done if limit.water is missing and add is TRUE.

The default line-drawing algorithm uses contourLines to obtain smooth-looking diagonal and curved lines, at the expense of not coinciding exactly with the rectangular grid that is used for drawing colors. lty, col, and lwd can be specified, but limiting the lines via xrange is not currently supported. Set lty.cr or lty.aq to 0 to suppress boundary lines between minerals or aqueous species.

To go back to the old behavior for drawing lines, set dotted to '0'. The old behavior does not respect lty; instead, the style of the boundary lines on 2-D diagrams can be altered by supplying one or more non-zero integers in dotted, which indicates the fraction of line segments to omit; a value of '1' or NULL for dotted has the effect of not drawing the boundary lines.

Activity Coefficients

The wording in this page and names of variables in functions refer exclusively to 'activities' of aqueous species. However, if activity coefficients are calculated (using the IS argument in affinity), then these variables are effectively transformed to molalities (see inst/tinytest/test-logmolality.

So that the labels on diagrams are adjusted accordingly, diagram sets the molality argument of axis.label to TRUE if IS was supplied as an argument to affinity. The labeling as molality takes effect even if IS is set to 0; this way, by including (or not) the IS = 0 argument to affinity, the user decides whether to label aqueous species variables as molality (or activity) for calculations at zero ionic strength (where molality = activity).

Other Functions

find.tp finds the locations in a matrix of integers that are surrounded by the greatest number of different values. The function counts the unique values in a 3x3 grid around each point and returns a matrix of indices (similar to which(..., arr.ind = TRUE)) for the maximum count (ties result in more than one pair of indices). It can be used with the output from diagram for calculations in 2 dimensions to approximately locate the triple points on the diagram.

Value

diagram returns an invisible list containing, first, the contents of eout, i.e. the output of affinity or equilibrate supplied in the function call. To this are added the names of the plotted variable in plotvar, the labels used for species (which may be plotmath expressions if format.names is TRUE) in names, and the values used for plotting in a list named plotvals. For 1-D diagrams, plotvals usually corresponds to the chemical activities of the species (i.e. eout$loga.equil), or, if alpha is TRUE, their mole fractions (degrees of formation). For 2-D diagrams, plotvals corresponds to the values of affinity (from eout$values) divided by the respective balancing coefficients for each species. For 2-D diagrams, the output also contains the matrices predominant, which identifies the predominant species in eout$species at each grid point, and predominant.values, which has the affinities of the predominant species divided by the balancing coefficients (if eout is the output of affinity) or the activities of the predominant species (if eout is the output of equilibrate). The rows and columns of these matrices correspond to the *x* and *y* variables, respectively.

References

Aksu, S. and Doyle, F. M. (2001) Electrochemistry of copper in aqueous glycine solutions. *J. Electrochem. Soc.* **148**, B51–B57.

Dick, J. M. (2019) CHNOSZ: Thermodynamic calculations and diagrams for geochemistry. *Front. Earth Sci.* **7**:180. doi:10.3389/feart.2019.00180

Dick, J. M. and Shock, E. L. (2011) Calculation of the relative chemical stabilities of proteins as a function of temperature and redox chemistry in a hot spring. *PLOS One* **6**, e22782. doi:10.1371/journal.pone.0022782

Helgeson, H. C. (1970) A chemical and thermodynamic model of ore deposition in hydrothermal systems. *Mineral. Soc. Amer. Spec. Pap.* **3**, 155–186. https://www.worldcat.org/oclc/583263

Helgeson, H. C., Delany, J. M., Nesbitt, H. W. and Bird, D. K. (1978) Summary and critique of the thermodynamic properties of rock-forming minerals. *Am. J. Sci.* **278-A**, 1–229. https://www.worldcat.org/oclc/13594862

LaRowe, D. E. and Helgeson, H. C. (2007) Quantifying the energetics of metabolic reactions in diverse biogeochemical systems: electron flow and ATP synthesis. *Geobiology* **5**, 153–168. doi:10.1111/j.14724669.2007.00099.x

Majzlan, J., Navrotsky, A., McClesky, R. B. and Alpers, C. N. (2006) Thermodynamic properties and crystal structure refinement of ferricopiapite, coquimbite, rhomboclase, and $Fe_2(SO_4)_3(H_2O)_5$. *Eur. J. Mineral.* **18**, 175–186. doi:10.1127/09351221/2006/00180175

Tagirov, B. and Schott, J. (2001) Aluminum speciation in crustal fluids revisited. *Geochim. Cosmochim. Acta* **65**, 3965–3992. doi:10.1016/S00167037(01)007050

See Also

Berman, mix, mosaic, nonideal, solubility, and util.plot are other help topics that use diagram in their examples. See the demos for even more examples.

Examples

```
## Calculate the equilibrium logarithm of activity of a
## basis species in different reactions
basis("CHNOS")
species(c("ethanol", "lactic acid", "deoxyribose", "ribose"))
a <- affinity(T = c(0, 150))
diagram(a, type = "O2", legend.x = "topleft", col = rev(rainbow(4)), lwd = 2)
title(main = "Equilibrium logfO2 for 1e-3 mol/kg of CO2 and ... ")

### 1-D diagrams: logarithms of activities

## Degrees of formation of ionized forms of glycine
## After Fig. 1 of Aksu and Doyle, 2001
basis("CHNOS+")
species(ispecies <- info(c("glycinium", "glycine", "glycinate")))
a <- affinity(pH = c(0, 14))
e <- equilibrate(a)
diagram(e, alpha = TRUE, lwd = 1)
title(main = paste("Degrees of formation of aqueous glycine species\n",
  "after Aksu and Doyle, 2001"))

## Degrees of formation of ATP species as a function of
## temperature, after LaRowe and Helgeson, 2007, Fig. 10b
# to make a similar diagram, activity of Mg+2 here is set to
# 10^-4, which is different from LH07, who used 10^-3 total molality
basis(c("CO2", "NH3", "H2O", "H3PO4", "O2", "H+", "Mg+2"),
  c(999, 999, 999, 999, 999, -5, -4))
species(c("HATP-3", "H2ATP-2", "MgATP-2", "MgHATP-"))
a <- affinity(T = c(0, 120, 25))
e <- equilibrate(a)
diagram(e, alpha = TRUE)
title(main = paste("Degrees of formation of ATP species,\n",
  "pH=5, log(aMg+2)=-3. After LaRowe and Helgeson, 2007"),
  cex.main = 0.9)

### 2-D diagrams: predominance diagrams
### These use the maximum affinity method

## Fe-S-O at 200 deg C, after Helgeson, 1970
basis(c("Fe", "oxygen", "S2"))
```

```
species(c("iron", "ferrous-oxide", "magnetite",
  "hematite", "pyrite", "pyrrhotite"))
# The calculations include the polymorphic transitions of
# pyrrhotite; no additional step is needed
a <- affinity(S2 = c(-50, 0), O2 = c(-90, -10), T=200)
diagram(a, fill = "heat")
title(main = paste("Fe-S-O, 200 degrees C, 1 bar",
  "After Helgeson, 1970", sep = "\n"))

## pe-pH diagram for hydrated iron sulfides,
## goethite and pyrite, after Majzlan et al., 2006
basis(c("Fe+2", "SO4-2", "H2O", "H+", "e-"),
  c(0, log10(3), log10(0.75), 999, 999))
species(c("rhomboclase", "ferricopiapite", "hydronium jarosite",
  "goethite", "melanterite", "pyrite"))
a <- affinity(pH = c(-1, 4, 256), pe = c(-5, 23, 256))
d <- diagram(a, main = "Fe-S-O-H, after Majzlan et al., 2006")
water.lines(d, lwd = 2)
text(3, 22, describe.basis(2:3, digits = 2, oneline = TRUE))
text(3, 21, describe.property(c("T", "P"), c(25, 1), oneline = TRUE))

## Aqueous Al species, after Tagirov and Schott, 2001
basis(c("Al+3", "F-", "H+", "O2", "H2O"))
AlOH <- c("Al(OH)4-", "Al(OH)3", "Al(OH)2+", "AlOH+2")
Al <- "Al+3"
AlF <- c("AlF+2", "AlF2+", "AlF3", "AlF4-")
AlOHF <- c("Al(OH)2F2-", "Al(OH)2F", "AlOHF2")
species(c(AlOH, Al, AlF, AlOHF), "aq")
res <- 300
a <- affinity(pH = c(0.5, 6.5, res), `F-` = c(-2, -9, res), T = 200)
diagram(a, fill = "terrain")
dprop <- describe.property(c("T", "P"), c(200, "Psat"))
legend("topright", legend = dprop, bty = "n")
mtitle(c("Aqueous aluminum species",
         "After Tagirov and Schott, 2001 Fig. 4d"), cex = 0.95)

## Temperature-Pressure: kayanite-sillimanite-andalusite
# cf. Fig. 49 of Helgeson et al., 1978
# this is a system of one component (Al2SiO5), however:
# - number of basis species must be the same as of elements
# - avoid using H2O or other aqueous species because of
#     T/P limits of the water() calculations;
basis(c("corundum", "quartz", "oxygen"))
species(c("kyanite", "sillimanite", "andalusite"))
# Database has transition temperatures of kyanite and andalusite
# at 1 bar only, so we permit calculation at higher temperatures
a <- affinity(T = c(200, 900, 99), P = c(0, 9000, 101), exceed.Ttr = TRUE)
d <- diagram(a, fill = NULL)
slab <- syslab(c("Al2O3", "SiO2", "H2O"))
mtitle(c(as.expression(slab), "after Helgeson et al., 1978"))
# Find the approximate position of the triple point
tp <- find.tp(d$predominant)
Ttp <- a$vals[[1]][tp[1, 2]]
```

```
Ptp <- rev(a$vals[[2]])[tp[1, 1]]
points(Ttp, Ptp, pch = 10, cex = 5)
```

`EOSregress` *Regress Equations-of-State Parameters for Aqueous Species*

Description

Fit experimental volumes and heat capacities using regression equations. Possible models include the Helgeson-Kirkham-Flowers (HKF) equations of state, or other equations defined using any combination of terms derived from the temperature, pressure and thermodynamic and electrostatic properties of water.

Usage

```
EOSregress(exptdata, var = "", T.max = 9999, ...)
EOSvar(var, T, P, ...)
EOScalc(coefficients, T, P, ...)
EOSplot(exptdata, var = NULL, T.max = 9999, T.plot = NULL,
  fun.legend = "topleft", coefficients = NULL, add = FALSE,
  lty = par("lty"), col=par("col"), ...)
EOSlab(var, coeff = "")
EOScoeffs(species, property, P=1)
Cp_s_var(T = 298.15, P = 1, omega.PrTr = 0, Z = 0)
V_s_var(T = 298.15, P = 1, omega.PrTr = 0, Z = 0)
```

Arguments

`exptdata`	dataframe, experimental data
`var`	character, name(s) of variables in the regression equations
`T.max`	numeric, maximum temperature for regression, in degrees Kelvin
`T`	numeric, temperature in Kelvin
`P`	numeric, pressure in bars
`...`	arguments specifying additional dependencies of the regression variables
`T.plot`	numeric, upper limit of temperature range to plot
`fun.legend`	character, where to place legend on plot
`coefficients`	dataframe, coefficients to use to make line on plot
`add`	logical, add lines to an existing plot?
`lty`	line style
`col`	color of lines
`coeff`	numeric, value of equation of state parameter for plot legend
`species`	character, name of aqueous species
`property`	character, '`Cp`' or '`V`'
`omega.PrTr`	numeric, value of omega at reference T and P
`Z`	numeric, charge

Details

EOSregress uses a linear model (lm) to regress the experimental heat capacity or volume data in exptdata, which is a data frame with columns 'T' (temperature in degrees Kelvin), 'P' (pressure in bars), and 'Cp' or 'V' (heat capacity in cal/mol.K or volume in cm3/mol). The 'Cp' or 'V' data must be in the third column. Only data below the temperature of T.max are included in the regression. The regression formula is specified by a vector of names in var. The names of the variables can be any combination of the following (listed in the order of search): variables listed in the following table, any available property of water (e.g. 'V', 'alpha', 'QBorn'), or the name of a function that can be found using get in the default environment. Examples of the latter are Cp_s_var, V_s_var, or functions defined by the user in the global environment; the arguments of these functions must include, but are not limited to, T and P.

T	T (temperature)
P	P (pressure)
TTheta	$(T - \Theta)$ (Θ = 228 K)
invTTheta	$1/(T - \Theta)$
TTheta2	$(T - \Theta)^2$
invTTheta2	$1/(T - \Theta)^2$
invPPsi	$1/(P + \Psi)$ (Ψ = 2600 bar)
invPPsiTTheta	$1/((P + \Psi)(T - \Theta))$
TXBorn	TX (temperature times X Born function)
drho.dT	$d\rho/dT$ (temperature derivative of density of water)
V.kT	$V\kappa_T$ (volume times isothermal compressibility of water)

EOSvar calculates the value of the variable named var (defined as described above) at the specified T (temperature in degrees Kelvin) and P (pressure in bars). This function is used by EOSregress to get the values of the variables used in the regression.

EOScalc calculates the predicted heat capacities or volumes using coefficients provided by the result of EOSregress, at the temperatures and pressures specified by T and P.

EOSplot takes a table of data in exptdata, runs EOSregress and EOScalc and plots the results. The experimental data are plotted as points, and the calculated values as a smooth line. The point symbols are filled circles where the calculated value is within 10% of the experimental value; open circles otherwise.

EOSlab produces labels for the variables listed above that can be used as.expressions in plots. The value of coeff is prefixed to the name of the variable (using substitute, with a multiplication symbol). For the properties listed in the table above, and selected properties listed in water, the label is formatted using plotmath expressions (e.g., with italicized symbols and Greek letters). If var is a user-defined function, the function can be given a 'label' attribute to provide plotmath-style formatting; in this case the appropriate multiplication or division symbol should be specified (see example below).

EOScoeffs retrieves coefficients in the Helgeson-Kirkham-Flowers equations from the thermodynamic database (thermo$OBIGT) for the given aqueous species. If the property is 'Cp', the resulting data frame has column names of '(Intercept)', 'invTTheta2' and 'TX', respectively holding the coefficients c_1, c_2 and ω in the equation $Cp^\circ = c_1 + c_2/(T - \Theta)^2 + \omega TX$. If the property is 'V', the data frame has column names of '(Intercept)', 'invTTheta' and 'Q', respectively holding the coefficients σ, ξ and ω in $V^\circ = \sigma + \xi/(T - \Theta) - \omega Q$. Here, σ

and ξ are calculated from a1, a2, a3 and a4 in `thermo()$OBIGT` at the pressure indicated by `P` (default 1 bar).

The original motivation for writing these functions was to explore alternatives or possible modifications to the revised Helgeson-Kirkham-Flowers equations applied to aqueous nonelectrolytes. As pointed out by Schulte et al., 2001, the functional forms of the equations do not permit retrieving values of the solvation parameter (ω) that closely represent the observed trends in both heat capacity and volume at high temperatures (above ca. 200 °C).

The examples below assume that the ω parameter in the HKF functions is a constant (does not depend on T and P), as is appropriate for nonelectrolytes. For charged species, the variables `Cp_s_var` and `V_s_var` can be used in the regressions. They correspond to the solvation contribution to heat capacity or volume, respectively, in the HKF EOS, divided by the value of ω at the reference temperature and pressure. Because these variables are themselves a function of `omega.PrTr`, an iterative procedure is needed to perform the regression.

Note that variables `QBorn` and `V_s_var` are both negated, so that the value of ω has its proper sign in the corresponding equations.

Value

For `EOSregress`, an object of class "lm". `EOSvar` and `EOScalc` both return numeric values. `EOScoeffs` returns a data frame.

References

Hnědkovský, L. and Wood, R. H. (1997) Apparent molar heat capacities of aqueous solutions of CH_4, CO_2, H_2S, and NH_3 at temperatures from 304 K to 704 K at a pressure of 28 MPa. *J. Chem. Thermodyn.* **29**, 731–747. doi:10.1006/jcht.1997.0192

Schulte, M. D., Shock, E. L. and Wood, R. H. (1995) The temperature dependence of the standard-state thermodynamic properties of aqueous nonelectrolytes. *Geochim. Cosmochim. Acta* **65**, 3919–3930. doi:10.1016/S00167037(01)007177

See Also

The vignette **eos-regress.html** has more references and examples, including an iterative method to retrieve `omega.PrTr`.

Examples

```
## Fit experimental heat capacities of CH4
## using revised Helgeson-Kirkham-Flowers equations
# Read the data from Hnedkovsky and Wood, 1997
f <- system.file("extdata/cpetc/HW97_Cp.csv", package = "CHNOSZ")
d <- read.csv(f)
# Use data for CH4
d <- d[d$species == "CH4", ]
d <- d[, -1]
# Convert J to cal and MPa to bar
d$Cp <- convert(d$Cp, "cal")
d$P <- convert(d$P, "bar")
# Specify the terms in the HKF equations
```

```
var <- c("invTTheta2", "TXBorn")
# Perform regression, with a temperature limit
EOSlm <- EOSregress(d, var, T.max = 600)
# Calculate the Cp at some temperature and pressure
EOScalc(EOSlm$coefficients, 298.15, 1)
# Get the database values of c1, c2 and omega for CH4(aq)
CH4coeffs <- EOScoeffs("CH4", "Cp")
## Make plots comparing the regressions
## with the accepted EOS parameters of CH4
opar <- par(mfrow = c(2,2))
EOSplot(d, T.max = 600)
title("Cp of CH4(aq), fit to 600 K")
legend("bottomleft", pch = 1, legend = "Hnedkovsky and Wood, 1997")
EOSplot(d, coefficients = CH4coeffs)
title("Cp from EOS parameters in database")
EOSplot(d, T.max = 600, T.plot = 600)
title("Cp fit to 600 K, plot to 600 K")
EOSplot(d, coefficients = CH4coeffs, T.plot = 600)
title("Cp from EOS parameters in database")
par(opar)

# Continuing from above, with user-defined variables
Theta <- 228  # K
invTTTheta3 <- function(T, P) (2*T) / (T-T*Theta) ^ 3
invTX <- function(T, P) 1 / T * water("XBorn", T = T, P = P)[,1]
# Print the calculated values of invTTTheta3
EOSvar("invTTTheta3", d$T, d$P)
# Use invTTTheta and invTX in a regression
var <- c("invTTTheta3", "invTX")
EOSregress(d, var)
# Give them a "label" attribute for use in the legend
attr(invTTTheta3, "label") <-
  quote(phantom()%*%2 * italic(T) / (italic(T) - italic(T) * Theta) ^ 3)
attr(invTX, "label") <- quote(phantom() / italic(T * X))
# Uncomment the following to make the plot
#EOSplot(d, var)

## Model experimental volumes of CH4
## using HKF equation and an exploratory one
f <- system.file("extdata/cpetc/HWM96_V.csv", package = "CHNOSZ")
d <- read.csv(f)
# Use data for CH4 near 280 bar
d <- d[d$species == "CH4", ]
d <- d[, -1]
d <- d[abs(d$P - 28) < 0.1, ]
d$P <- convert(d$P, "bar")
# The HKF equation
varHKF <- c("invTTheta", "QBorn")
# alpha is the expansivity coefficient of water
varal <- c("invTTheta", "alpha")
opar <- par(mfrow = c(2, 2))
# For both HKF and the expansivity equation,
# we'll fit up to a temperature limit
```

```
EOSplot(d, varHKF, T.max = 663, T.plot = 625)
legend("bottomright", pch = 1, legend = "Hnedkovsky et al., 1996")
title("V of CH4(aq), HKF equation")
EOSplot(d, varal, T.max = 663, T.plot = 625)
title("V of CH4(aq), expansivity equation")
EOSplot(d, varHKF, T.max = 663)
title("V of CH4(aq), HKF equation")
EOSplot(d, varal, T.max = 663)
title("V of CH4(aq), expansivity equation")
par(opar)
# Note that the volume regression using the HKF gives
# a result for omega (coefficient on Q) that is
# not consistent with the high-T heat capacities
```

`equilibrate` *Equilibrium Chemical Activities of Species*

Description

Calculate equilibrium chemical activities of species from the affinities of formation of the species at unit activity.

Usage

```
equilibrate(aout, balance = NULL, loga.balance = NULL,
  ispecies = !grepl("cr", aout$species$state), normalize = FALSE,
  as.residue = FALSE, method = c("boltzmann", "reaction"),
  tol = .Machine$double.eps^0.25)
equil.boltzmann(Astar, n.balance, loga.balance)
equil.reaction(Astar, n.balance, loga.balance, tol = .Machine$double.eps^0.25)
moles(eout)
```

Arguments

`aout`	list, output from `affinity`
or `mosaic`	
`balance`	character or numeric, how to balance the transformations
`ispecies`	numeric, which species to include
`normalize`	logical, normalize the molar formulas of species by the balancing coefficients?
`as.residue`	logical, report results for the normalized formulas?
`Astar`	numeric, affinities of formation reactions excluding species contribution
`n.balance`	numeric, number of moles of balancing component in the formation reactions of the species of interest
`loga.balance`	numeric (single value or vector), logarithm of total activity of balanced quantity
`method`	character, equilibration method to use
`tol`	numeric, convergence tolerance for `uniroot`
`eout`	list, output from `equilibrate`

Details

equilibrate calculates the chemical activities of species in metastable equilibrium, for constant temperature, pressure and chemical activities of basis species, using specified balancing constraints on reactions between species.

It takes as input aout, the output from affinity, giving the chemical affinities of formation reaction of each species, which may be calculated on a multidimensional grid of conditions. Alternatively, aout can be the output from mosaic, in which case the equilibrium activities of the formed species are calculated and combined with those of the changing basis species to make an object that can be plotted with diagram.

The equilibrium chemical activities of species are calculated using either the equil.reaction or equil.boltzmann functions, the latter only if the balance is on one mole of species.

equilibrate needs to be provided constraints on how to balance the reactions representing transformations between the species. balance indicates the balancing component, according to the following scheme:

- 'NULL': autoselect
- *name of basis species*: balance on this basis species
- 'length': balance on length of proteins
- '1': balance on one mole of species
- *numeric vector*: user-defined constraints

The default value of NULL for balance indicates to use the coefficients on the basis species that is present (i.e. with non-zero coefficients) in all formation reactions, or if that fails, to set the balance to '1'. However, if all the species (as listed in code aout$species) are proteins (have an underscore character in their names), the default value of NULL for balance indicates to use 'length' as the balance.

NOTE: The summation of activities assumes an ideal system, so 'molality' is equivalent to 'activity' here. loga.balance gives the logarithm of the total activity of balance (which is total activity of species for '1' or total activity of amino acid residue-equivalents for 'length'). If loga.balance is missing, its value is taken from the activities of species listed in aout; this default is usually the desired operation. The supplied value of loga.balance may also be a vector of values, with length corresponding to the number of conditions in the calculations of affinity.

normalize if TRUE indicates to normalize the molar formulas of species by the balance coefficients. This operation is intended for systems of proteins, whose conventional formulas are much larger than the basis speices. The normalization also applies to the balancing coefficients, which as a result consist of '1's. After normalization and equilibration, the equilibrium activities are then re-scaled (for the original formulas of the species), unless as.residue is TRUE.

equil.boltzmann is used to calculate the equilibrium activities if balance is '1' (or when normalize or as.residue is TRUE), otherwise equil.reaction is called. The default behavior can be overriden by specifying either 'boltzmann' or 'reaction' in method. Using equil.reaction may be needed for systems with huge (negative or positive) affinities, where equil.boltzmann produces a NaN result.

ispecies can be supplied to identify a subset of the species to include in the equilibrium calculation. By default, this is all species except solids (species with 'cr' state). However, the stability regions of solids are still calculated (by a call to diagram without plotting). At all points outside

of their stability region, the logarithms of activities of solids are set to -999. Likewise, where any solid species is calculated to be stable, the logarithms of activities of all aqueous species are set to -999.

`moles` simply calculates the total number of moles of elements corresponding to the activities of formed species in the output from `equilibrate`.

Value

`equil.reaction` and `equil.boltzmann` each return a list with dimensions and length equal to those of `Astar`, giving the `log10` of the equilibrium activities of the species of interest. `equilibrate` returns a list, containing first the values in `aout`, to which are appended `m.balance` (the balancing coefficients if `normalize` is TRUE, a vector of '`1`'s otherwise), `n.balance` (the balancing coefficients if `normalize` is FALSE, a vector of '`1`'s otherwise), `loga.balance`, `Astar`, and `loga.equil` (the calculated equilibrium activities of the species).

Algorithms

The input values to `equil.reaction` and `equil.boltzmann` are in a list, `Astar`, all elements of the list having the same dimensions; they can be vectors, matrices, or higher-dimensionsal arrays. Each list element contains the chemical affinities of the formation reactions of one of the species of interest (in dimensionless base-10 units, i.e. A/2.303RT), calculated at unit activity of the species of interest. The equilibrium base-10 logarithm activities of the species of interest returned by either function satisfy the constraints that 1) the final chemical affinities of the formation reactions of the species are all equal and 2) the total activity of the balancing component is equal to (`loga.balance`). The first constraint does *not* impose a complete equilibrium, where the affinities of the formation reactions are all equal to zero, but allows for a metastable equilibrium, where the affinities of the formation reactions are equal to each other.

In `equil.reaction` (the algorithm described in Dick, 2008 and the only one available prior to CHNOSZ_0.8), the calculations of relative abundances of species are based on a solving a system of equations representing the two constraints stated above. The solution is found using `uniroot` with a flexible method for generating initial guesses.

In `equil.boltzmann`, the chemical activities of species are calculated using the Boltzmann distribution. This calculation is faster than the algorithm of `equil.reaction`, but is limited to systems where the transformations are all balanced on one mole of species. If `equil.boltzmann` is called with `balance` other than '`1`', it stops with an error.

Warning

Despite its name, this function does not generally produce a complete equilibrium. It returns activities of species such that the affinities of formation reactions are equal to each other (and transformations between species have zero affinity); this is a type of metastable equilibrium. Although they are equal to each other, the affinities are not necessarily equal to zero. Use `solubility` to find complete equilibrium, where the affinities of the formation reactions become zero.

References

Dick, J. M. (2008) Calculation of the relative metastabilities of proteins using the CHNOSZ software package. *Geochem. Trans.* **9**:10. doi:10.1186/14674866910

See Also

diagram has examples of using equilibrate to make equilibrium activity diagrams. palply is used by both equil.reaction and equil.boltzmann to parallelize intensive parts of the calculations.

See the vignette **multi-metal.html** for an example of balancing on two elements (N in the basis species, C in the formed species).

Examples

```
## Equilibrium in a simple system:
## ionization of carbonic acid
basis("CHNOS+")
species(c("CO2", "HCO3-", "CO3-2"))
# Set unit activity of the species (0 = log10(1))
species(1:3, 0)
# Calculate Astar (for unit activity)
res <- 100
Astar <- affinity(pH = c(0, 14, res))$values
# The logarithms of activity for a total activity
# of the balancing component (CO2) equal to 0.001
loga.boltz <- equil.boltzmann(Astar, c(1, 1, 1), 0.001)
# Calculated another way
loga.react <- equil.reaction(Astar, c(1, 1, 1), rep(0.001, 100))
# They should be pretty close
stopifnot(all.equal(loga.boltz, loga.react))
# The first ionization constant (pKa)
ipKa <- which.min(abs(loga.boltz[[1]] - loga.boltz[[2]]))
pKa.equil <- seq(0, 14, length.out = res)[ipKa]
# Calculate logK directly
logK <- subcrt(c("CO2","H2O","HCO3-","H+"), c(-1, -1, 1, 1), T = 25)$out$logK
# We could decrease tolerance here by increasing res
stopifnot(all.equal(pKa.equil, -logK, tolerance = 1e-2))
```

---

examples *Run Examples from the Documentation*

Description

Run the examples contained in each of the documentation topics.

Usage

```
examples(save.png = FALSE)
demos(which = c("sources", "protein.equil", "affinity", "NaCl",
  "density", "ORP", "ionize", "buffer", "protbuff",
  "glycinate", "mosaic", "copper", "arsenic", "solubility", "gold",
  "contour", "sphalerite", "minsol", "Shh", "saturation",
  "adenine", "DEW", "lambda", "potassium", "TCA", "aluminum", "AD",
  "comproportionation", "Pourbaix", "E_coli", "yttrium", "rank.affinity"),
  save.png = FALSE)
```

Arguments

save.png	logical, generate PNG image files for the plots?
which	character, which example to run

Details

examples runs all the examples in the help pages for the package. example is called for each topic with ask set to FALSE (so all of the figures are shown without prompting the user).

demos runs all the demos in the package. The demo(s) to run is/are specified by which; the default is to run them in the order of the list below.

sources Cross-check the reference list with the thermodynamic database

protein.equil Chemical activities of two proteins in metastable equilibrium (Dick and Shock, 2011)

affinity Affinities of metabolic reactions and amino acid synthesis (Amend and Shock, 1998, 2001)

NaCl Equilibrium constant for aqueous NaCl dissociation (Shock et al., 1992)

density Density of H_2O, inverted from IAPWS-95 equations (rho.IAPWS95)

ORP Temperature dependence of oxidation-reduction potential for redox standards

ionize ionize.aa(): contour plots of net charge and ionization properties of LYSC_CHICK

buffer Minerals and aqueous species as buffers of hydrogen fugacity (Schulte and Shock, 1995)

protbuff Chemical activities buffered by thiol peroxidases or sigma factors

glycinate Metal-glycinate complexes (Shock and Koretsky, 1995; Azadi et al., 2019)

mosaic Eh-pH diagram with two sets of changing basis species (Garrels and Christ, 1965)

copper Another example of mosaic: complexation of Cu with glycine (Aksu and Doyle, 2001)

arsenic Another example of mosaic: Eh-pH diagram for the system As-O-H-S (Lu and Zhu, 2011)

solubility Solubility of calcite (cf. Manning et al., 2013) and CO_2 (cf. Stumm and Morgan, 1996)

gold Solubility of gold (Akinfiev and Zotov; 2001; Stefánsson and Seward, 2004; Williams-Jones et al., 2009)

contour Gold solubility contours on a log fO2 - pH diagram (Williams-Jones et al., 2009)

sphalerite Solubility of sphalerite (Akinfiev and Tagirov, 2014)

minsol Solubilities of multiple minerals

dehydration $\log K$ of dehydration reactions; SVG file contains tooltips and links

Shh Affinities of transcription factors relative to Sonic hedgehog (Dick, 2015)

saturation Equilibrium activity diagram showing activity ratios and mineral saturation limits (Bowers et al., 1984)

adenine HKF regression of heat capacity and volume of aqueous adenine (Lowe et al., 2017)

DEW Deep Earth Water (DEW) model for high pressures (Sverjensky et al., 2014a and 2014b)

lambda Effects of lambda transition on thermodynamic properties of quartz (Berman, 1988)

potassium Comparison of thermodynamic datasets for predicting mineral stabilities (Sverjensky et al., 1991)

TCA Standard Gibbs energies of the tricarboxylic (citric) acid cycle (Canovas and Shock, 2016)

aluminum Reactions involving Al-bearing minerals (Zimmer et al., 2016; Tutolo et al., 2014)

carboxylase Rank abundance distribution for RuBisCO and acetyl-CoA carboxylase

AD Dissolved gases: Henry's constant, volume, and heat capacity (Akinfiev and Diamond, 2003)

comproportionation Gibbs energy of sulfur comproportionation (Amend et al., 2020)

Pourbaix Eh-pH diagram for Fe-O-H with equisolubility lines (Pourbaix, 1974)

E_coli Gibbs energy of biomass synthesis in *E. coli* (LaRowe and Amend, 2016)

rank.affinity Affinity ranking for proteins in yeast nutrient limitation (data from Tai et al., 2005)

yttrium `logB.to.OBIGT` fits at 800 and 1000 bar and Y speciation in `NaCl` solution at varying pH (Guan et al., 2020)

For either function, if `save.png` is TRUE, the plots are saved in `png` files whose names begin with the names of the help topics or demos.

Two of the demos have external dependencies and are not automatically run by `demos`. '`dehydration`' creates an interactive SVG file; this demo depends on **RSVGTipsDevice**, which is not available for Windows. '`carboxylase`' creates an animated GIF; this demo requires that the ImageMagick `convert` commmand be available on the system (tested on Linux and Windows).

'`carboxylase`' animates diagrams showing rankings of calculated chemical activities along a combined T and $\log a_{H_2}$ gradient, or makes a single plot on the default device (without conversion to animated GIF) if a single temperature (`T`) is specified in the code. To run this demo, an empty directory named '`png`' must be present (as a subdirectory of the R working directory). The proteins in the calculation are 24 carboxylases from a variety of organisms. There are 12 ribulose phosphate carboxylase and 12 acetyl-coenzyme A carboxylase; 6 of each type are from nominally mesophilic organisms and 6 from nominally thermophilic organisms, shown as blue and red symbols on the diagrams. The activities of hydrogen at each temperature are calculated using $\log a_{H_{2(aq)}} = -11 + 3/(40 \times T\,(°C))$; this equation comes from a model of relative stabilities of proteins in a hot-spring environment (Dick and Shock, 2011).

Warning

The discontinuities apparent in the plot made by the `NaCl` demo illustrate limitations of the "g function" for charged species in the revised HKF model (the 355 °C boundary of region II in Figure 6 of Shock et al., 1992). Note that SUPCRT92 (Johnson et al., 1992) gives similar output at 500 bar. However, SUPCRT does not output thermodynamic properties above 350 °C at P_{SAT}; see Warning in `subcrt`.

References

Akinfiev, N. N. and Diamond, L. W. (2003) Thermodynamic description of aqueous nonelectrolytes at infinite dilution over a wide range of state parameters. *Geochim. Cosmochim. Acta* **67**, 613–629. doi:10.1016/S00167037(02)011419

Akinfiev, N. N. and Tagirov, B. R. (2014) Zn in hydrothermal systems: Thermodynamic description of hydroxide, chloride, and hydrosulfide complexes. *Geochem. Int.* **52**, 197–214. doi:10.1134/S0016702914030021

Akinfiev, N. N. and Zotov, A. V. (2001) Thermodynamic description of chloride, hydrosulfide, and hydroxo complexes of Ag(I), Cu(I), and Au(I) at temperatures of 25-500°C and pressures of 1-2000 bar. *Geochem. Int.* **39**, 990–1006.

Aksu, S. and Doyle, F. M. (2001) Electrochemistry of copper in aqueous glycine solutions. *J. Electrochem. Soc.* **148**, B51–B57.

Amend, J. P. and Shock, E. L. (1998) Energetics of amino acid synthesis in hydrothermal ecosystems. *Science* **281**, 1659–1662. doi:10.1126/science.281.5383.1659

Amend, J. P. and Shock, E. L. (2001) Energetics of overall metabolic reactions of thermophilic and hyperthermophilic Archaea and Bacteria. *FEMS Microbiol. Rev.* **25**, 175–243. doi:10.1016/S01686445(00)000620

Amend, J. P., Aronson, H. S., Macalady, J. and LaRowe, D. E. (2020) Another chemolithotrophic metabolism missing in nature: sulfur comproportionation. *Environ. Microbiol.* **22**, 1971–1976. doi:10.1111/14622920.14982

Azadi, M. R., Karrech, A., Attar, M. and Elchalakani, M. (2019) Data analysis and estimation of thermodynamic properties of aqueous monovalent metal-glycinate complexes. *Fluid Phase Equilib.* **480**, 25-40. doi:10.1016/j.fluid.2018.10.002

Berman, R. G. (1988) Internally-consistent thermodynamic data for minerals in the system Na_2O-K_2O-CaO-MgO-FeO-Fe_2O_3-Al_2O_3-SiO_2-TiO_2-H_2O-CO_2. *J. Petrol.* **29**, 445-522. doi:10.1093/petrology/29.2.445

Bowers, T. S., Jackson, K. J. and Helgeson, H. C. (1984) *Equilibrium Activity Diagrams for Coexisting Minerals and Aqueous Solutions at Pressures and Temperatures to 5 kb and 600°C*, Springer-Verlag, Berlin, 397 p. https://www.worldcat.org/oclc/11133620

Canovas, P. A., III and Shock, E. L. (2016) Geobiochemistry of metabolism: Standard state thermodynamic properties of the citric acid cycle. *Geochim. Cosmochim. Acta* **195**, 293–322. doi:10.1016/j.gca.2016.08.028

Dick, J. M. and Shock, E. L. (2011) Calculation of the relative chemical stabilities of proteins as a function of temperature and redox chemistry in a hot spring. *PLOS One* **6**, e22782. doi:10.1371/journal.pone.0022782

Dick, J. M. (2015) Chemical integration of proteins in signaling and development. *bioRxiv*. doi:10.1101/015826

Garrels, R. M. and Christ, C. L. (1965) *Solutions, Minerals, and Equilibria*, Harper & Row, New York, 450 p. https://www.worldcat.org/oclc/517586

Guan, Q., Mei, Y., Etschmann, B., Testemale, D., Louvel, M. and Brugger, J. (2020) Yttrium complexation and hydration in chloride-rich hydrothermal fluids: A combined *ab initio* molecular dynamics and *in situ* X-ray absorption spectroscopy study. *Geochim. Cosmochim. Acta* **281**, 168–189. doi:10.1016/j.gca.2020.04.015

Johnson, J. W., Oelkers, E. H. and Helgeson, H. C. (1992) SUPCRT92: A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bar and 0 to 1000°C. *Comp. Geosci.* **18**, 899–947. doi:10.1016/0098-3004(92)90029Q

LaRowe, D. E. and Amend, J. P. (2016) The energetics of anabolism in natural settings. *ISME J.* **10**, 1285–1295. doi:10.1038/ismej.2015.227

Lowe, A. R., Cox, J. S. and Tremaine, P. R. (2017) Thermodynamics of aqueous adenine: Standard partial molar volumes and heat capacities of adenine, adeninium chloride, and sodium adeninate from T = 278.15 K to 393.15 K. *J. Chem. Thermodyn.* **112**, 129–145. doi:10.1016/j.jct.2017.04.005

Lu, P. and Zhu, C. (2011) Arsenic Eh–pH diagrams at 25°C and 1 bar. *Environ. Earth Sci.* **62**, 1673–1683. doi:10.1007/s126650100652x

Manning, C. E., Shock, E. L. and Sverjensky, D. A. (2013) The chemistry of carbon in aqueous fluids at crustal and upper-mantle conditions: Experimental and theoretical constraints. *Rev. Mineral. Geochem.* **75**, 109–148. doi:10.2138/rmg.2013.75.5

Pourbaix, M. (1974) *Atlas of Electrochemical Equilibria in Aqueous Solutions*, NACE, Houston, TX and CEBELCOR, Brussels. https://www.worldcat.org/oclc/563921897

Schulte, M. D. and Shock, E. L. (1995) Thermodynamics of Strecker synthesis in hydrothermal systems. *Orig. Life Evol. Biosph.* **25**, 161–173. doi:10.1007/BF01581580

Shock, E. L. and Koretsky, C. M. (1995) Metal-organic complexes in geochemical processes: Estimation of standard partial molal thermodynamic properties of aqueous complexes between metal cations and monovalent organic acid ligands at high pressures and temperatures. *Geochim. Cosmochim. Acta* **59**, 1497–1532. doi:10.1016/00167037(95)000588

Shock, E. L., Oelkers, E. H., Johnson, J. W., Sverjensky, D. A. and Helgeson, H. C. (1992) Calculation of the thermodynamic properties of aqueous species at high pressures and temperatures: Effective electrostatic radii, dissociation constants and standard partial molal properties to 1000 °C and 5 kbar. *J. Chem. Soc. Faraday Trans.* **88**, 803–826. doi:10.1039/FT9928800803

Stefánsson, A. and Seward, T. M. (2004) Gold(I) complexing in aqueous sulphide solutions to 500°C at 500 bar. *Geochim. Cosmochim. Acta* **68**, 4121–4143. doi:10.1016/j.gca.2004.04.006

Stumm, W. and Morgan, J. J. (1996) *Aquatic Chemistry: Chemical Equilibria and Rates in Natural Waters*, John Wiley & Sons, New York, 1040 p. https://www.worldcat.org/oclc/31754493

Sverjensky, D. A., Harrison, B. and Azzolini, D. (2014a) Water in the deep Earth: The dielectric constant and the solubilities of quartz and corundum to 60 kb and 1,200 °C. *Geochim. Cosmochim. Acta* **129**, 125–145. doi:10.1016/j.gca.2013.12.019

Sverjensky, D. A., Hemley, J. J. and D'Angelo, W. M. (1991) Thermodynamic assessment of hydrothermal alkali feldspar-mica-aluminosilicate equilibria. *Geochim. Cosmochim. Acta* **55**, 989-1004. doi:10.1016/00167037(91)90157Z

Sverjensky, D. A., Stagno, V. and Huang, F. (2014b) Important role for organic carbon in subduction-zone fluids in the deep carbon cycle. *Nat. Geosci.* **7**, 909–913. doi:10.1038/ngeo2291

Tai, S. L., Boer, V. M., Daran-Lapujade, P., Walsh, M. C., de Winde, J. H., Daran, J.-M. and Pronk, J. T. (2005) Two-dimensional transcriptome analysis in chemostat cultures: Combinatorial effects of oxygen availability and macronutrient limitation in *Saccharomyces cerevisiae*. *J. Biol. Chem.* **280**, 437–447. doi:10.1074/jbc.M410573200

Tutolo, B. M., Kong, X.-Z., Seyfried, W. E., Jr. and Saar, M. O. (2014) Internal consistency in aqueous geochemical data revisited: Applications to the aluminum system. *Geochim. Cosmochim. Acta* **133**, 216–234. doi:10.1016/j.gca.2014.02.036

Williams-Jones, A. E., Bowell, R. J. and Migdisov, A. A. (2009) Gold in solution. *Elements* **5**, 281–287. doi:10.2113/gselements.5.5.281

Zimmer, K., Zhang, Y., Lu, P., Chen, Y., Zhang, G., Dalkilic, M. and Zhu, C. (2016) SUPCRTBL: A revised and extended thermodynamic dataset and software package of SUPCRT92. *Comp. Geosci.* **90**, 97–111. doi:10.1016/j.cageo.2016.02.013

Examples

```
demos(c("ORP", "NaCl"))
```

`extdata` *Extra Data*

Description

The files in the subdirectories of `extdata` provide additional thermodynamic data and other data to support the examples in the package documentation and vignettes. See `thermo` for a description of the files in `extdata/OBIGT`, which are used to generate the thermodynamic database.

Details

Files in `Berman` contain thermodynamic data for minerals using the Berman formulation:

- `Ber88_1988.csv` contains thermodynamic data for minerals taken from Berman (1988).
- Other files with names like `xxx_yyyy.csv` contain thermodynamic data from other sources; xxx in the filename corresponds to the reference in `thermo$OBIGT` and yyyy gives the year of publication. `Berman` uses these data for the calculation of thermodynamic properties at specified P and T, which are then available for use in `subcrt`. If there are any duplicated mineral names in the files, only the most recent data are used, as determined by the year in the file name. Following conventions used SUPCRT92 (see Helgeson et al., 1978), the names of sanidine and microcline were changed to K-feldspar,high and K-feldspar,low (by using the same names in all data files, loading the optional SUPCRT92 data file updates these minerals rather than makes new ones).
- `sympy.R` is an R script that uses **rSymPy** to symbolically integrate Bermans's equations for heat capacity and volume to write experessions for enthalpy, entropy and Gibbs energy.
- The `testing` directory contains data files based on Berman and Aranovich (1996). These are used to demonstrate the addition of data from a user-supplied file (see `Berman`).

Files in `cpetc` contain experimental and calculated thermodynamic and environmental data:

- `PM90.csv` Heat capacities of four unfolded aqueous proteins taken from Privalov and Makhatadze, 1990. Temperature in °C is in the first column, and heat capacities of the proteins in J mol^{-1} K^{-1} in the remaining columns. See `ionize.aa` and the vignette **anintro.html** for examples that use this file.
- `RH95.csv` Heat capacity data for iron taken from Robie and Hemingway, 1995. Temperature in Kelvin is in the first column, heat capacity in J K^{-1} mol^{-1} in the second. See `subcrt` for an example that uses this file.
- `SOJSH.csv` Experimental equilibrium constants for the reaction NaCl(aq) = Na+ + Cl- as a function of temperature and pressure taken from Fig. 1 of Shock et al., 1992. See `demo("NaCl")` for an example that uses this file.

- `HWM96_V.csv`, `HW97_Cp.csv` Apparent molar volumes and heat capacities of CH_4, CO_2, H_2S, and NH_3 in dilute aqueous solutions reported by Hnědkovský et al., 1996 and Hnědkovský and Wood, 1997. Units are Kelvin, MPa, J/K/mol, and cm3/mol. See `demo("AD")`, `EOSregress` and the vignette **eos-regress.html** for examples that use these files.
- `SC10_Rainbow.csv` Values of temperature (°C), pH and logarithms of activity of CO_2, H_2, NH_4^+, H_2S and CH_4 for mixing of seawater and hydrothermal fluid at Rainbow field (Mid-Atlantic Ridge), taken from Shock and Canovas, 2010. See the vignette **anintro.html** for an example that uses this file.
- `SS98_Fig5a.csv`, `SS98_Fig5b.csv` Values of logarithm of fugacity of O_2 and pH as a function of temperature for mixing of seawater and hydrothermal fluid, digitized from Figs. 5a and b of Shock and Schulte, 1998. See the vignette **anintro.html** for an example that uses this file.
- `rubisco.csv` UniProt IDs for Rubisco, ranges of optimal growth temperature of organisms, domain and name of organisms, and URL of reference for growth temperature, from Dick, 2014. See `rank.affinity` and the vignette **anintro.html** for examples that use this file.
- `bluered.txt` Blue - light grey - red color palette, computed using **colorspace**: `::diverge_hcl(1000, c = 100, l = c(50, 90), power = 1)`. This is used by `ZC.col`.
- `AD03_Fig1?.csv` Experimental data points digitized from Figure 1 of Akinfiev and Diamond, 2003, used in `demo("AD")`.
- `TKSS14_Fig2.csv` Experimental data points digitized from Figure 2 of Tutolo et al., 2014, used in `demo("aluminum")`.
- `Mer75_Table4.csv` Values of log(aK+/aH+) and log(aNa+/aH+) from Table 4 of Merino, 1975, used in `demo("aluminum")`.

Files in `protein` contain protein sequences and amino acid compositions for proteins.

- `EF-Tu.aln` consists of aligned sequences (394 amino acids) of elongation factor Tu (EF-Tu). The sequences correspond to those taken from UniProtKB for ECOLI (*Escherichia coli*), THETH (*Thermus thermophilus*) and THEMA (*Thermotoga maritima*), and reconstructed ancestral sequences taken from Gaucher et al., 2003 (maximum likelihood bacterial stem and mesophilic bacterial stem, and alternative bacterial stem). See `read.fasta` for an example that uses this file.
- `rubisco.fasta` Sequences of Rubisco obtained from UniProt (see Dick, 2014). See the vignette **anintro.html** for an example that uses this file.
- `POLG.csv` Amino acid compositions of a few proteins used for some tests and examples. These are various subunits of the Poliovirus type 1 polyprotein (POLG_POL1M in UniProt).
- `TBD+05.csv` lists genes with transcriptomic expression changes in carbon limitation stress response experiments in yeast (Tai et al., 2005).
- `TBD+05_aa.csv` has the amino acid compositions of proteins coded by those genes. The last two files are used in `demo{"rank.affinity"}`.

Files in `taxonomy` contain taxonomic data files:

- `names.dmp` and `nodes.dmp` are excerpts of NCBI taxonomy files (`https://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz`, accessed 2010-02-15). These files contain only the entries for *Escherichia coli* K-12, *Saccharomyces cerevisiae*, *Homo sapiens*, *Pyrococcus furisosus* and *Methanocaldococcus jannaschii* (taxids 83333, 4932, 9606, 186497,

243232) and the higher-ranking nodes (genus, family, etc.) in the respective lineages. See `taxonomy` for examples that use these files.

Files in `adds` contain additional thermodynamic data and group additivity definitions:

- `BZA10.csv` contains supplementary thermodynamic data taken from Bazarkina et al. (2010). The data can be added to the database in the current session using `add.OBIGT`. See `add.OBIGT` for an example that uses this file.
- `OBIGT_check.csv` contains the results of running `check.OBIGT` to check the internal consistency of entries in the default and optional datafiles.
- `RH98_Table15.csv` Group stoichiometries for high molecular weight crystalline and liquid organic compounds taken from Table 15 of Richard and Helgeson, 1998. The first three columns have the `compound` name, `formula` and physical `state` ('`cr`' or '`liq`'). The remaining columns have the numbers of each group in the compound; the names of the groups (columns) correspond to species in `thermo$OBIGT`. The compound named '`5a(H),14a(H)-cholestane`' in the paper has been changed to '`5a(H),14b(H)-cholestane`' here to match the group stoichiometry given in the table. See `RH2OBIGT` for a function that uses this file.
- `SK95.csv` contains thermodynamic data for alanate, glycinate, and their complexes with metals, taken from Amend and Helgeson (1997) and Shock and Koretsky (1995) as corrected in slop98.dat. These data are used in `demo("copper")` and `demo("glycinate")`.
- `LA19_test.csv` contains thermodynamic data for dimethylamine and trimethylamine from LaRowe and Amend (2019) in energy units of both J and cal. This file is used in `test-util.data.R`) to check the messages produced by `check.GHS` and `check.EOS`.

References

Akinfiev, N. N. and Diamond, L. W. (2003) Thermodynamic description of aqueous nonelectrolytes at infinite dilution over a wide range of state parameters. *Geochim. Cosmochim. Acta* **67**, 613–629. doi:10.1016/S00167037(02)011419

Amend, J. P. and Helgeson, H. C. (1997) Calculation of the standard molal thermodynamic properties of aqueous biomolecules at elevated temperatures and pressures. Part 1. L-α-amino acids. *J. Chem. Soc., Faraday Trans.* **93**, 1927–1941. doi:10.1039/A608126F

Bazarkina, E. F., Zotov, A. V. and Akinfiev, N. N. (2010) Pressure-dependent stability of cadmium chloride complexes: Potentiometric measurements at 1–1000 bar and 25°C. *Geol. Ore Deposits* **52**, 167–178. doi:10.1134/S1075701510020054

Berman, R. G. (1988) Internally-consistent thermodynamic data for minerals in the system Na_2O-K_2O-CaO-MgO-FeO-Fe_2O_3-Al_2O_3-SiO_2-TiO_2-H_2O-CO_2. *J. Petrol.* **29**, 445-522. doi:10.1093/petrology/29.2.445

Berman, R. G. and Aranovich, L. Ya. (1996) Optimized standard state and solution properties of minerals. I. Model calibration for olivine, orthopyroxene, cordierite, garnet, and ilmenite in the system FeO-MgO-CaO-Al_2O_3-TiO_2-SiO_2. *Contrib. Mineral. Petrol.* **126**, 1-24. doi:10.1007/s004100050233

Dick, J. M. (2014) Average oxidation state of carbon in proteins. *J. R. Soc. Interface* **11**, 20131095. doi:10.1098/rsif.2013.1095

Gattiker, A., Michoud, K., Rivoire, C., Auchincloss, A. H., Coudert, E., Lima, T., Kersey, P., Pagni, M., Sigrist, C. J. A., Lachaize, C., Veuthey, A.-L., Gasteiger, E. and Bairoch, A. (2003) Automatic

annotation of microbial proteomes in Swiss-Prot. *Comput. Biol. Chem.* **27**, 49–58. doi:10.1016/S14769271(02)000944

Gaucher, E. A., Thomson, J. M., Burgan, M. F. and Benner, S. A (2003) Inferring the palaeoenvironment of ancient bacteria on the basis of resurrected proteins. *Nature* **425**(6955), 285–288. doi:10.1038/nature01977

Helgeson, H. C., Delany, J. M., Nesbitt, H. W. and Bird, D. K. (1978) Summary and critique of the thermodynamic properties of rock-forming minerals. *Am. J. Sci.* **278-A**, 1–229. https://www.worldcat.org/oclc/13594862

Hnědkovský, L., Wood, R. H. and Majer, V. (1996) Volumes of aqueous solutions of CH_4, CO_2, H_2S, and NH_3 at temperatures from 298.15 K to 705 K and pressures to 35 MPa. *J. Chem. Thermodyn.* **28**, 125–142. doi:10.1006/jcht.1996.0011

Hnědkovský, L. and Wood, R. H. (1997) Apparent molar heat capacities of aqueous solutions of CH_4, CO_2, H_2S, and NH_3 at temperatures from 304 K to 704 K at a pressure of 28 MPa. *J. Chem. Thermodyn.* **29**, 731–747. doi:10.1006/jcht.1997.0192

Joint Genome Institute (2007) Bison Pool Environmental Genome. Protein sequence files downloaded from IMG/M (https://img.jgi.doe.gov/)

LaRowe, D. E. and Amend, J. P. (2019) The energetics of fermentation in natural settings. *Geomicrobiol. J.* **36**, 492–505. doi:10.1080/01490451.2019.1573278

Merino, E. (1975) Diagenesis in teriary sandstones from Kettleman North Dome, California. II. Interstitial solutions: distribution of aqueous species at 100°C and chemical relation to diagenetic mineralogy. *Geochim. Cosmochim. Acta* **39**, 1629–1645. doi:10.1016/00167037(75)90085X

Privalov, P. L. and Makhatadze, G. I. (1990) Heat capacity of proteins. II. Partial molar heat capacity of the unfolded polypeptide chain of proteins: Protein unfolding effects. *J. Mol. Biol.* **213**, 385–391. doi:10.1016/S00222836(05)801986

Richard, L. and Helgeson, H. C. (1998) Calculation of the thermodynamic properties at elevated temperatures and pressures of saturated and aromatic high molecular weight solid and liquid hydrocarbons in kerogen, bitumen, petroleum, and other organic matter of biogeochemical interest. *Geochim. Cosmochim. Acta* **62**, 3591–3636. doi:10.1016/S00167037(97)003451

Robie, R. A. and Hemingway, B. S. (1995) *Thermodynamic Properties of Minerals and Related Substances at 298.15 K and 1 Bar (*10^5 *Pascals) Pressure and at Higher Temperatures*. U. S. Geol. Surv., Bull. 2131, 461 p. https://www.worldcat.org/oclc/32590140

Shock, E. and Canovas, P. (2010) The potential for abiotic organic synthesis and biosynthesis at seafloor hydrothermal systems. *Geofluids* **10**, 161–192. doi:10.1111/j.14688123.2010.00277.x

Shock, E. L. and Koretsky, C. M. (1995) Metal-organic complexes in geochemical processes: Estimation of standard partial molal thermodynamic properties of aqueous complexes between metal cations and monovalent organic acid ligands at high pressures and temperatures. *Geochim. Cosmochim. Acta* **59**, 1497–1532. doi:10.1016/00167037(95)000588

Shock, E. L., Oelkers, E. H., Johnson, J. W., Sverjensky, D. A. and Helgeson, H. C. (1992) Calculation of the thermodynamic properties of aqueous species at high pressures and temperatures: Effective electrostatic radii, dissociation constants and standard partial molal properties to 1000 °C and 5 kbar. *J. Chem. Soc. Faraday Trans.* **88**, 803–826. doi:10.1039/FT9928800803

Shock, E. L. and Schulte, M. D. (1998) Organic synthesis during fluid mixing in hydrothermal systems. *J. Geophys. Res.* **103**, 28513–28527. doi:10.1029/98JE02142

Tai, S. L., Boer, V. M., Daran-Lapujade, P., Walsh, M. C., de Winde, J. H., Daran, J.-M. and Pronk, J. T. (2005) Two-dimensional transcriptome analysis in chemostat cultures: Combinatorial effects of oxygen availability and macronutrient limitation in *Saccharomyces cerevisiae*. *J. Biol. Chem.* **280**, 437–447. doi:10.1074/jbc.M410573200

Tutolo, B. M., Kong, X.-Z., Seyfried, W. E., Jr. and Saar, M. O. (2014) Internal consistency in aqueous geochemical data revisited: Applications to the aluminum system. *Geochim. Cosmochim. Acta* **133**, 216–234. doi:10.1016/j.gca.2014.02.036

`IAPWS95` *Properties of Water from IAPWS-95*

Description

Calculate thermodynamic properties of water following the IAPWS-95 formulation.

Usage

```
IAPWS95(property, T = 298.15, rho = 1000)
```

Arguments

`property`	character, name(s) of property(s) to calculate
`T`	numeric, temperature (K)
`rho`	numeric, density (kg m^{-3})

Details

`IAPWS95` provides an implementation of the IAPWS-95 formulation for properties (including pressure) calculated as a function of temperature and density.

The `IAPWS95` function returns values of thermodynamic properties in specific units (per gram). The IAPWS-95 formulation follows the triple point convention used in engineering (values of internal energy and entropy are taken to be zero at the triple point).

For `IAPWS95` the upper temperature limit of validity is 1000 °C, but extrapolation to much higher temperatures is possible (Wagner and Pruss, 2002). Valid pressures are from the greater of zero bar or the melting pressure at temperature to 10000 bar (with the provision for extrapolation to more extreme conditions). The function does not check these limits and will attempt calculations for any range of input parameters, but may return `NA` for properties that fail to be calculated at given temperatures and pressures and/or produce warnings or even errors when problems are encountered.

Value

A data frame the number of rows of which corresponds to the number of input temperature, pressure and/or density values.

References

Wagner, W. and Pruss, A. (2002) The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *J. Phys. Chem. Ref. Data* **31**, 387–535. doi:10.1063/1.1461829

See Also

`util.water` for properties along the saturation curve (`WP02.auxiliary`) and calculation of density from pressure and temperature (`rho.IAPWS92`). `water.IAPWS95` is a wrapper around `IAPWS95` and the utility functions, which converts the specific units to molar quantities, and is used in higher-level functions (`water`).

Examples

```
# Calculate pressure for given temperature and density
IAPWS95("P", T = 500, rho = 838.0235)
```

---

info *Search the Thermodynamic Database*

Description

Search for species by name or formula, retrieve their thermodynamic properties and parameters, and add proteins to the thermodynamic database.

Usage

```
info(species = NULL, state = NULL, check.it=TRUE)
```

Arguments

`species`	character, names or formulas of species, or (for `info` only) numeric with same meaning as `ispecies`
`state`	character, physical states of the species
`check.it`	logical, check GHS and EOS parameters for self-consistency?

Details

`info` is the primary function used for querying the thermodynamic database (`thermo()$OBIGT`). It is often called recursively; first with a character value (or values) for `species` indicating the name(s) or formula(s) of the species of interest. The result of this call is a numeric value, which can be provided as an argument in a second call to `info` in order to retrieve a data frame of the thermodynamic properties of the species.

The text of `species` is searched in the names, chemical formulas, and abbreviations (in the 'abbrv' column) in the thermodynamic database. If the text of the `species` is matched, the index of that species is returned. If there are multiple matches for the `species`, and `state` is

NULL, the index of first match is returned. The order of entries in the database is grouped by states in the order 'aq', 'cr', 'gas', 'liq'. Therefore, for substances represented by both aqueous and gaseous species, the index of the aqueous species is returned, unless state is set to 'gas'. Note that names (not formulas) of inorganic species, such as 'oxygen' and 'methane', are used only for the gas.

Names of species including an underscore character are indicative of proteins, e.g. 'LYSC_CHICK'. If the name of a protein is provided to info and the composition of the protein can be found using pinfo, the thermodyamic properties and parameters of the nonionized protein (calculated using amino acid group additivity) are added to the thermodynamic database. Included in the return value, as for other species, is the index of the protein in the thermodynamic database or NA if the protein is not found. Names of proteins and other species can be mixed.

If no exact matches are found, info searches the database for similar names or formulas using agrep. If any of these are found, the results are summarized on the screen, but the function always returns NA in this case.

With a numeric argument, the rows of thermo()$OBIGT indicated by ispecies are returned, after removing any order-of-magnitude scaling factors (see thermo). If these species are all aqueous or are all not aqueous, the compounded column names used in thermo()$OBIGT are replaced with names appropriate for the corresponding equations of state. A missing value of one of the standard molal Gibbs energy (G) or enthalpy (H) of formation from the elements or entropy (S) is calculated from the other two, if available. If check.it is TRUE, several checks of self consistency among the thermodynamic properties and parameters are performed using check.GHS and check.EOS.

See Also

retrieve for searching species by element; check.OBIGT for checking self-consistency of each species.

Examples

```
## Summary of available data
info()

## Species information
# Search for something named (or whose formula is) "Fe"
si <- info("Fe")
# Use the number to get the full entry
info(si)
# Show data for the higher-temperature phases
info(si:(si+3))

## Dealing with states
# Order of precedence for names:
# aq > cr > gas > liq
info(c("ethanol", "adenosine"))  # aq, aq
# State argument overrides the default
info(c("ethanol", "adenosine"), state = c("gas", "cr"))
# Exceptions: gases have precedence for names of methane and inorganic gases
info(c("methane", "oxygen")) # gas, gas
```

```
# Formulas default to aqueous species, if available
i1 <- info(c("CH4", "CO2", "CS2", "MgO"))
info(i1)$state  # aq, aq, gas, cr
# State argument overrides the default
i2 <- info(c("CH4", "CO2", "MgO"), "gas")
info(i2)$state  # gas, gas, NA

## Partial name or formula searches
info("ATP")
info("thiol")
info("MgC")
# Add an extra character to refine a search
# or to search using terms that have exact matches
info("MgC ")
info("acetate ")
info(" H2O")
```

`ionize.aa` *Properties of Ionization of Proteins*

Description

Calculate the charges of proteins and contributions of ionization to the thermodynamic properties of proteins.

Usage

```
ionize.aa(aa, property = "Z", T = 25, P = "Psat", pH = 7,
  ret.val = NULL, suppress.Cys = FALSE)
```

Arguments

`aa`	data frame, amino acid composition in the format of `thermo()$protein`
`property`	character, property to calculate
`T`	numeric, temperature in °C
`P`	numeric, pressure in bar, or '`Psat`' for vapor pressure of H_2O above 100 °C
`pH`	numeric, pH
`ret.val`	character, return the indicated value from intermediate calculations
`suppress.Cys`	logical, suppress (ignore) the ionization of the cysteine groups?

Details

The properties of ionization of proteins calculated by this function take account of the standard molal thermodynamic properties of ionizable amino acid sidechain groups and the terminal groups in proteins ([AABB]) and their equations of state parameters taken from Dick et al., 2006. The values of the ionization constants (pK) are calculated as a function of temperature, and the charges and the ionization contributions of other thermodynamic properties to the proteins are calculated additively,

without consideration of electrostatic interactions, so they are best applied to the unfolded protein reference state.

For each amino acid composition in `aa`, the additive value of the `property` is calculated as a function of `T`, `P` and `pH`. `property` can be NULL to denote net charge, or if not NULL is one of the properties available in `subcrt`, or is 'A' to calculate the dimensionless chemical affinity (A/2.303RT) of the ionization reaction for the protein. If `ret.val` is one of '`pK`', '`alpha`', or '`aavals`' it indicates to return the value of the ionization constant, degree of formation, or the values of the `property` for each ionizable group rather than taking their sums for the amino acid compositions in `aa`.

Value

The function returns a matrix (possibly with only one row or column) with number of rows corresponding to the longest of `T`, `P` or `pH` (values of any of these with shorter length are recycled) and a column for each of the amino acid compositions in `aa`.

References

Dick, J. M., LaRowe, D. E. and Helgeson, H. C. (2006) Temperature, pressure, and electrochemical constraints on protein speciation: Group additivity calculation of the standard molal thermodynamic properties of ionized unfolded proteins. *Biogeosciences* **3**, 311–336. doi:10.5194/bg33112006

Makhatadze, G. I. and Privalov, P. L. (1990) Heat capacity of proteins. 1. Partial molar heat capacity of individual amino acid residues in aqueous solution: Hydration effect. *J. Mol. Biol.* **213**, 375–384. doi:10.1016/S00222836(05)801974

Privalov, P. L. and Makhatadze, G. I. (1990) Heat capacity of proteins. II. Partial molar heat capacity of the unfolded polypeptide chain of proteins: Protein unfolding effects. *J. Mol. Biol.* **213**, 385–391. doi:10.1016/S00222836(05)801986

See Also

`pinfo`, `affinity`

Examples

```
## Heat capacity of LYSC_CHICK as a function of T
pH <- c(5, 9, 3)
T <- seq(0, 100)
# Cp of non-ionized protein
Cp.nonion <- subcrt("LYSC_CHICK", T = T)$out[[1]]$Cp
plot(T, Cp.nonion, xlab = axis.label("T"), type = "l",
  ylab = axis.label("Cp"), ylim = c(20000, 35000))
# Cp of ionization and ionized protein
aa <- pinfo(pinfo("LYSC_CHICK"))
for(pH in c(5, 9, 3)) {
  Cp.ionized <- Cp.nonion + ionize.aa(aa, "Cp", T = T, pH = pH)[, 1]
  lines(T, Cp.ionized, lty = 2)
  text(80, Cp.ionized[70], paste("pH =", pH) )
}
# Makhatadze and Privalov's group contributions
T <- c(5, 25, 50, 75, 100, 125)
```

```
points(T, MP90.cp("LYSC_CHICK", T))
# Privalov and Makhatadze's experimental values
e <- read.csv(system.file("extdata/cpetc/PM90.csv", package = "CHNOSZ"))
points(e$T, e$LYSC_CHICK, pch = 16)
legend("bottomright", pch = c(16, 1, NA, NA), lty = c(NA, NA, 1, 2),
  legend = c("PM90 experiment", "MP90 groups",
  "DLH06 groups no ion", "DLH06 groups ionized"))
title("Heat capacity of unfolded LYSC_CHICK")
```

`logB.to.OBIGT` *Fit Thermodynamic Parameters to Formation Constants (*$\log \beta$*)*

Description

Fit thermodynamic parameters to experimental formation constants for an aqueous species and add the parameters to OBIGT.

Usage

```
logB.to.OBIGT(logB, species, coeffs, T, P, npar = 3,
  optimize.omega = FALSE, tolerance = 0.05)
```

Arguments

`logB`	Values of $\log \beta$
`species`	Species in the formation reaction (the formed species must be last)
`coeffs`	Coefficients of the formation reaction
`T`	Temperature (units of `T.units`)
`P`	Temperature (‘`Psat`’ or numerical values with units of `P.units`)
`npar`	numeric, number of parameters to use in fitting
`optimize.omega`	logical, optimize the omega parameter (relevant for charged species)?
`tolerance`	Tolerance for checking equivalence of input and calculated $\log \beta$ values

Details

This function updates the `OBIGT` thermodynamic database with parameters fit to formation constants of aqueous species as a function of temperature. The `logB` argument should have decimal logarithm of formation constants for an aqueous complex ($\log \beta$). The formation reaction is defined by `species` and `coeffs`. All species in the formation reaction must be present in OBIGT except for the *last* species, which is the newly formed species.

The function works as follows. First, the properties of the known species are combined with $\log \beta$ to calculate the standard Gibbs energy (G[T]) of the formed species at each value of `T` and `P`. Then, `lm` is used to fit one or more of the thermodynamic parameters `G`, `S`, `c1`, `c2`, and `omega` to the values of G[T]. The first two of these parameters are standard-state values at 25 °C and 1 bar, and the last three are parameters in the revised HKF equations (e.g. Eq. B25 of Shock et al., 1992). The

fitted parameters for the formed species are then added to OBIGT. Finally, `all.equal` is used to test for approximate equivalence of the input values of $\log \beta$ and calculated equilibrium constants; if the mean absolute difference exceeds `tolerance`, an error occurs.

To avoid overfitting, only the first three parameters (`G`, `S`, and `c1`) are used by default. More parameters (up to 5) or fewer (down to 1) can be selected by changing `npar`. Volumetric parameters (a_1 to a_4) in the HKF equations currently aren't included, so the resulting fits are valid only at the input pressure values.

Independent of `npar`, the number of parameters used in the fit is not more than the number of data values (n). If n is less than 5, then the values of the unused parameters are set to 0 for addition to OBIGT. For instance, a single value of $\log \beta$ would be fit only with `G`, implying that computed values of G[T] have no temperature dependence.

The value of ω is a constant in the revised HKF equations for uncharged species, but for charged species, it is a function of T and P as described by the "g function" (Shock et al., 1992). An optimization step is available to refine the value of `omega` at 25 °C and 1 bar for charged species taking its temperature dependence into account for the fitting. However, representative datasets are not well represented by variable `omega` (see first example), so this step is skipped by default. Furthermore, `logB.to.OBIGT` by default also sets the `z` parameter in OBIGT to 0; this enforces a constant ω for charged species in calculations with `subcrt` (note that this is a parameter for the HKF equations and does not affect reaction balancing). Set `optimize.omega` to TRUE to override the defaults and activate variable ω for charged species; this takes effect only if `npar == 5` and $n > 5$.

Value

The species index of the new species in OBIGT.

References

Migdisov, Art. A., Zezin, D. and Williams-Jones, A. E. (2011) An experimental study of Cobalt (II) complexation in Cl^- and H_2S-bearing hydrothermal solutions. *Geochim. Cosmochim. Acta* **75**, 4065–4079. doi:10.1016/j.gca.2011.05.003

Mei, Y., Sherman, D. M., Liu, W., Etschmann, B., Testemale, D. and Brugger, J. (2015) Zinc complexation in chloride-rich hydrothermal fluids (25–600 °C): A thermodynamic model derived from *ab initio* molecular dynamics. *Geochim. Cosmochim. Acta* **150**, 264–284. doi:10.1016/j.gca.2014.09.023

Shock, E. L., Oelkers, E. H., Johnson, J. W., Sverjensky, D. A. and Helgeson, H. C. (1992) Calculation of the thermodynamic properties of aqueous species at high pressures and temperatures: Effective electrostatic radii, dissociation constants and standard partial molal properties to 1000 °C and 5 kbar. *J. Chem. Soc. Faraday Trans.* **88**, 803–826. doi:10.1039/FT9928800803

See Also

`logB.to.OBIGT` calls `mod.OBIGT` with `zap = TRUE` to clear the parameters of a formed species if it already exists in the OBIGT database. If preexisting parameters (e.g. volumetric HKF coefficients) weren't cleared, they would interfere with the fitting done here, which uses only selected parameters.

Examples

```
## CoHS+ from Migdisov et al. (2011)
logB <- c(6.24, 6.02, 5.84, 5.97, 6.52)
T <- c(120, 150, 200, 250, 300)
P <- "Psat"
species <- c("Co+2", "HS-", "CoHS+")
coeffs <- c(-1, -1, 1)
opar <- par(mfrow = c(2, 1))
for(o.o in c(TRUE, FALSE)) {
  # Fit the parameters with or without variable omega
  inew <- logB.to.OBIGT(logB, species, coeffs, T, P, npar = 5, optimize.omega = o.o)
  # Print the new database entry
  info(inew)
  # Plot experimental logB
  plot(T, logB, "n", c(100, 320), c(5.8, 6.8),
    xlab = axis.label("T"), ylab = quote(log~beta))
  points(T, logB, pch = 19, cex = 2)
  # Plot calculated values
  Tfit <- seq(100, 320, 10)
  sres <- subcrt(species, coeffs, T = Tfit)
  lines(sres$out$T, sres$out$logK, col = 4)
  title(describe.reaction(sres$reaction))
  legend <- c("Migdisov et al. (2011)",paste0("logB.to.OBIGT(optimize.omega = ",o.o,")"))
  legend("top", legend, pch = c(19, NA), lty = c(0, 1), col = c(1, 4),
    pt.cex = 2, bg = "#FFFFFFB0")
}
par(opar)
# NB. Optimizing omega leads to unphysical oscillations in the logK (first plot)

## ZnCl+ from Mei et al. (2015)
# Values for 5000 bar calculated with the modified Ryzhenko-Bryzgalin (RB) model
logB <- c(-1.93,-1.16,-0.38,0.45,1.15,1.76,2.30,2.80,3.26,3.70,4.12,4.53,4.92)
species <- c("Zn+2", "Cl-", "ZnCl+")
coeffs <- c(-1, -1, 1)
T <- c(25, 60, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600)
P <- 5000
# Note: ZnCl+ is in the default OBIGT database;
# logB.to.OBIGT() "zaps" the previous parameters before adding the fitted ones
inew <- logB.to.OBIGT(logB, species, coeffs, T, P, npar = 5)
# Plot RB and logB.to.OBIGT values
plot(T, logB, xlab = axis.label("T"), ylab = axis.label("logB"), pch = 19, cex = 2)
Tfit <- seq(25, 600, 25)
sres <- subcrt(species, coeffs, T = Tfit, P = P)
lines(sres$out$T, sres$out$logK, col = 4)
title(describe.reaction(sres$reaction), line = 3)
title("5000 bar", font.main = 1, line = 1)
legend <- c("Mei et al. (2015)", "logB.to.OBIGT()")
legend("topleft", legend, pch = c(19, NA), lty = c(0, 1), col = c(1, 4), pt.cex = 2)
```

---

makeup *Parse Chemical Formulas*

Description

Count the elements and charges in a chemical formula.

Usage

```
makeup(formula, multiplier = 1, sum = FALSE, count.zero = FALSE)
count.elements(formula)
```

Arguments

formula	character, a chemical formula
multiplier	numeric, multiplier for the elemental counts in each formula
sum	logical, add together the elemental counts in all formulas?
count.zero	logical, include zero counts for elements?

Details

makeup parses a chemical formula expressed in string notation, returning the numbers of each element in the formula. The formula may carry a charge, indicated by a + or - sign, possibly followed by a magnitude, after the uncharged part of the formula. The formula may have multiple subformulas enclosed in parentheses (but the parentheses may not be nested), each one optionally followed by a numeric coefficient. The formula may have one suffixed subformula, separated by '*' or ':', optionally preceded by a numeric coefficient. All numbers may contain a decimal point.

Each subformula (or the entire formula without subformulas) should be a simple formula. A simple formula, processed by count.elements, must adhere to the following pattern: it starts with an elemental symbol; all elemental symbols start with an uppercase letter, and are followed by another elemental symbol, a number (possibly fractional, possibly signed), or nothing (the end of the formula). Any sequence of one uppercase letter followed by zero or more lowercase letters is recognized as an elemental symbol. makeup will issue a warning for elemental symbols that are not present in thermo$element.

makeup can handle numeric and length > 1 values for the formula argument. If the argument is numeric, it identifies row number(s) in thermo()$OBIGT from which to take the formulas of species. If formula has length > 1, the function returns a list containing the elemental counts in each of the formulas. If count.zero is TRUE, the elemental counts for each formula include zeros to indicate elements that are only present in any of the other formulas.

The multiplier argument must have either length = 1 or length equal to the number of formulas. The elemental count in each formula is multiplied by the respective value. If sum is true, the elemental counts in all formulas (after any multiplying) are summed together to yield a single bulk formula.

Value

A numeric vector with names refering to each of the elemental symbols in the formula. If more than one formula is provided, a list of numeric vectors is returned, unless sum is TRUE.

See Also

mass, entropy, basis, i2A

Examples

```
# Elemental composition of a simple compound
makeup("CO2")     # 1 carbon, 2 oxygen
# Formula of lawsonite, with a parenthetical part and a suffix
makeup("CaAl2Si2O7(OH)2*H2O")
# Fractional coefficients are OK
reddiv10 <- makeup("C10.6N1.6P0.1")
10*reddiv10 # 106, 16, 1 (Redfield ratio)

# The coefficient for charge is a number with a *preceding* sign
# e.g., ferric iron, with a charge of +3 is expressed as
makeup("Fe+3")
# Transcribing the formula the way it appears in many
# publications produces a likely unintended result:
# 3 iron atoms and a charge of +1
makeup("Fe3+")

# These all represent a single negative charge, i.e., electron
makeup("-1")
makeup("Z-1+0")
makeup("Z0-1")   # the "old" formula for the electron in thermo()$OBIGT
makeup("(Z-1)")  # the current formula in thermo()$OBIGT

# Hypothetical compounds with negative numbers of elements
makeup("C-4(O-2)")   # -4 carbon, -2 oxygen
makeup("C-4O-2")     # -4 carbon,  1 oxygen, -2 charge
makeup("C-4O-2-2")   # -4 carbon, -2 oxygen, -2 charge

# The 'sum' argument can be used to check mass and charge
# balance in a chemical reaction
formula <- c("H2O", "H+", "(Z-1)", "O2")
mf <- makeup(formula, c(-1, 2, 2, 0.5), sum = TRUE)
all(mf == 0)  # TRUE
```

`mix` *Combine Diagrams for Multi-Metal Systems*

Description

Combine diagrams for different systems by mashing or rebalancing two diagrams or mixing two diagrams with a third.

Usage

```
mash(d1, d2)
rebalance(d1, d2, balance = NULL)
mix(d1, d2, d3, parts = c(1, 1), .balance = NULL)
```

Arguments

d1	list, output of diagram for the first mono-metallic system
d2	list, output of diagram for the second mono-metallic system
balance	character or numeric, specification of secondary balancing coefficients
d3	list, output of diagram for the bimetallic system
parts	numeric, amount of each metal (i.e. fixed composition) for the mixed system
.balance	*argument for internal use only*

Details

These functions make a new affinity object from the output of diagram. The result can be used to make a new diagram that shows the combined system.

mash creates a set of intersecting predominance fields for all possible combinations of species in d1 and d2. The new names are formed from the names used in the source diagrams; for example if "Cp" and "Py" are predominant minerals at the same position in diagrams 1 and 2, the field for the mashed diagram will be labeled "Cp+Py". The affinities are calculated by summing the formation reactions from the two diagrams to give equal parts of the balancing coefficients in d1 and d2 (that is, equal parts of two different metals). Note that the actual values of the affinities (and therefore the ratio between the metals) doesn't affect the resulting diagram because the affinities are assigned values of -Inf wherever one of the species is not predominant in the respective single-metal diagram.

mix is an expanded form of mash that allows combinations not only between two single-metal diagrams (d1 and d2) but also between each of those diagrams and third diagram for bimetallic species (d3). All combinations of species in all crosses between the diagrams (d1–d2, d1–d3, d2–d3, d3–d3) are identified. The mole fractions of species in each combination are computed to satisfy the ratio of metals defined in parts. For example, if d1 and d2 are balanced on Fe^{+2} and VO_4^{-3}, the species are combined by default to give equal parts of Fe and V. Note that pairs of bimetallic species in d3 are included as well as single bimetallic species that satisfy the composition in parts (e.g. FeV for c(1, 1) or Fe_3V for c(3, 1)).

From the possible combinations of species, combinations are removed that have a negative mole fraction of any species or that involve any mono-metallic species that has no predominance field in the corresponding single-metal diagram. The output consists of each unique combination of species, including the combined formation reactions and affinities (in the species and values elements of the output list), The affinities are assigned values of -Inf wherever one of the species is not predominant in the respective single-metal diagram. Therefore, either the single-metal diagrams (d1 or d2) can be recovered by setting parts to c(1, 0) or c(0, 1), respectively.

NOTE: Unlike the diagram calls used to make d1 and d2, which by themselves should produce reasonable diagrams for a single-metal system, the d3 diagram by itself probably has no useful interpretation. It is only used in mix as a way to transmit the results of affinity for the bimetmal system and the formatted names that are made by diagram.

rebalance creates a new set of affinities of reactions *between* species in both systems. Diagrams for different systems likely use different primary balancing coefficients, such as balancing on different metals. rebalance uses *secondary* balancing coefficients, specified acording to balance (see equilibrate for a description of this argument), to determine the reactions between the species in the two systems. The affinities of these reactions are then used *only* to identify the predominant species at each grid point. The *returned* value of affinity are carried forward from those

used to make the source diagrams ('plotvals' in d1 and d2), and therefore reflect the primary balancing coefficients. The returned values are assigned -Inf wherever that species is determined to not predominate according to the secondary balancing.

Because mash yields finite values of affinity for only a single species at any grid point, the final diagram can be made with any setting of balance. mix gives combinations of species that each have the amount of metals defined in parts, so it makes no difference whether the final diagram is balanced on either of the metals, or on formula units (balance = 1). However, for rebalance, balance in the final diagram should be set to '1' to balance on formula units in order to preserve the primary balancing coefficients.

Value

A list object with the same structure as the output from affinity, so it can be used as input to diagram.

See Also

More examples are in the vignette **multi-metal.html**.

Examples

```
par(mfrow = c(2, 2))
# Define basis species with Fe and Cu
basis(c("Fe+2", "Cu+", "hydrogen sulfide", "oxygen", "H2O", "H+"))
xlab <- ratlab("Fe+2", "Cu+")
# Calculate diagram for only Fe-bearing minerals
species(c("pyrite", "pyrrhotite", "magnetite", "hematite"))
aFe <- affinity("Fe+2" = c(0, 12), O2 = c(-40, -16), T = 400, P = 2000)
dFe <- diagram(aFe, xlab = xlab, main = "Fe-S-O-H")
# Calculate diagram for only Cu-bearing minerals
species(c("covellite", "chalcocite", "tenorite", "cuprite"))
aCu <- affinity(aFe)  # argument recall
dCu <- diagram(aCu, xlab = xlab, main = "Cu-S-O-H")
### mash() diagram
ac <- mash(dFe, dCu)
diagram(ac, xlab = xlab, main = "Cu-Fe-S-O-H with mash()")
### rebalance() diagram
ad <- rebalance(dFe, dCu)
diagram(ad, xlab = xlab, balance = 1, main = "Cu-Fe-S-O-H with rebalance()")
db <- describe.basis(3)
leg <- lex(lTP(400, 2000), db)
legend("bottomleft", legend = leg, bty = "n")
```

---

mosaic *Chemical Affinities with Changing Basis Species*

Description

Calculate chemical affinities of formation reactions of species using basis species that change with the conditions.

Usage

```
mosaic(bases, blend = TRUE, stable = list(), loga_aq = NULL, ...)
```

Arguments

bases	character, basis species to be changed in the calculation, or list, each element of which defines an independent group of changing basis species
blend	logical, use relative abundances of basis species?
stable	list, previously determined stable species
loga_aq	numeric, activities of aqueous species (overrides current values in basis)
...	additional arguments to be passed to affinity

Details

mosaic calculates the affinities of formation of species when the relative abundances of the basis species listed in bases change over the range of conditions, due to e.g. ionization, complexation or redox reactions. This is a way to "speciate the basis species". For example, the speciation of sulfur ('SO4-2', 'HSO4-', 'HS-' and 'H2S') as a function of Eh and pH affects the formation affinities, and therefore relative stabilities of iron oxide and sulfide minerals. Chemical activity diagrams constructed by assembling sub-diagrams corresponding to the predominant (i.e. most stable) basis species can described as "mosaic diagrams".

The function calculates the affinities using all combination of basis species given as a vector argument to bases. Or, a list can be provided contain any number of vectors that specify different groups of basis species. All combinations of basis species in these groups are used for the calculations.

The first species listed in each group should be in the current basis definition, and all the basis species in each group should be related to the first basis species there (i.e. all share the same element). The arguments in ... are passed to affinity to specify the variable conditions, such as temperature, pressure, and activities of other basis species.

blend can be a single logical value or multiple values, in order to control the calculations for individual groups of basis species. If blend is TRUE (the default), the relative abundances of the basis species in each group are calculated using equilibrate, with the total activity taken from the corresponding basis species in the incoming basis definition. Then, the function calculates overall affinities of the formation reactions of each species by combining reactions written using individual basis species in proportion to the relative abundances of the basis species.

If blend is FALSE, the function returns the affinities calculated using the single predominant basis species in bases at each condition. This is appropriate when minerals, rather than aqueous species, are used as the changing basis species. Note, however, that mosaic is not internally recursive: the stabilities of one group of basis species (e.g. minerals) are not affected by changes in another group (e.g. aqueous species).

("Stacked mosaic diagrams") are useful for making diagrams for multi-metal systems. By using the stable minerals in one calculation as the new basis species in a subsequent calculation, a series of stacked mosaic diagrams with increasing complexity can be made. Specifically, this is done by supplying previously calculated stabilities (from the predominant element of the output of diagram) as an element of the list in the stable argument whose position corresponds to the appropriate group of basis species. Note that a value in any position of the stable list forces

blend = FALSE for the corresponding group of basis species, so there is no need to explicity change the blend argument.

The activities of mosaiced basis species in each group are taken from the current basis definition. Generally it makes sense to set the activity of minerals to 1 (logact = 0) and the activity of aqueous species to some smaller value. For mosaic stacking calculations where the mosaiced basis species include both minerals and aqueous species, the loga_aq argument specifies the activity of aqueous species to be used *in each group*. That is, there should be one value of loga_aq for each group of basis species; use NA to indicate that the activity comes from the current basis definition. See the Mosaic Stacking 2 section of the vignette **multi-metal.html** for an example.

Value

A list containing A.species (affinities of formation of the species with changing basis species) and A.bases (affinities of formation of the basis species in terms of the first basis species), each having same structure as the list returned by affinity. If bases is a list, A.bases is also a list, each element of which corresponds to a group of basis species in bases. If blend is TRUE, the output also contains E.bases (the output of equilibrate for each group of basis species)

References

Garrels, R. M. and Christ, C. L. (1965) *Solutions, Minerals, and Equilibria*, Harper & Row, New York, 450 p. https://www.worldcat.org/oclc/517586

See Also

demo("mosaic"), which extends the example below with carbonate species in order to plot a siderite field. To calculate mineral solubilities with mosaic calculations that account for ligand speciation, use bases as the first argument to solubility. stack_mosaic implements calculations for bimetallic systems.

Examples

```
# Fe-minerals and aqueous species in Fe-S-O-H system
# Speciate SO4-2, HSO4-, HS-, and H2S as a function of Eh and pH
# After Garrels and Christ, 1965 Figure 7.20
pH <- c(0, 14)
Eh <- c(-1, 1)
T <- 25
basis(c("FeO", "SO4-2", "H2O", "H+", "e-"))
basis("SO4-2", -6)
species(c("Fe+2", "Fe+3"), -6)
species(c("pyrrhotite", "pyrite", "hematite", "magnetite"), add = TRUE)
# The basis species we'll swap through
bases <- c("SO4-2", "HSO4-", "HS-", "H2S")
# Calculate affinities using the relative abundances of the basis species
# NOTE: set blend = FALSE for sharp transitions between the basis species
# (looks more like the diagram in GC65)
m1 <- mosaic(bases, pH = pH, Eh = Eh, T = T)
# Make a diagram and add water stability lines
d <- diagram(m1$A.species, lwd = 2)
water.lines(d, col = "seagreen", lwd = 1.5)
```

```
# Show lines for Fe(aq) = 10^-4 M
species(c("Fe+2", "Fe+3"), -4)
m2 <- mosaic(bases, pH = pH, Eh = Eh, T = T)
diagram(m2$A.species, add = TRUE, names = FALSE)
title(main=paste("Iron oxides and sulfides in water, log(total S) = -6",
  "After Garrels and Christ, 1965", sep="\n"))
legend("bottomleft", c("log(act_Fe) = -4", "log(act_Fe) = -6"), lwd = c(2, 1), bty = "n")
# We could overlay the predominance fields for the basis species
#diagram(m1$A.bases, add = TRUE, col = "blue", col.names = "blue", lty = 3)
```

`NaCl` *Simple NaCl-Water Solution*

Description

Calculate speciation and ionic strength of aqueous solutions with a given molality of NaCl.

Usage

```
NaCl(m_tot = 1, T = 25, P = "Psat", pH = NA, attenuate = FALSE)
```

Arguments

`m_tot`	numeric, total molality of NaCl (single value)
`T`	numeric, temperature in °C
`P`	numeric, pressure in bar
`pH`	numeric, pH
`attenuate`	logical, halve changes of variables in each step?

Details

Thermodynamic models for metal solubility and speciation involving chloride complexes are commonly specified in terms of amount of NaCl rather than activity (or molality) of Cl^- as an independent variable. This function calculates distribution of species and ionic strength in a simple aqueous solution given a total amount (`m_tot`, in mol/kg) of NaCl. The aqueous Cl-bearing species considered in the system are Cl^-, NaCl, and optionally HCl. Na^+ is present as a basis species, but the formation of Na-bearing species such as NaOH is not considered. The activity coefficients of charged species are calculated using the extended Debye-Hückel equation (see `nonideal`) via the `IS` argument of `affinity`. The function first sets the molality of Na^+ and ionic strength equal to `m_tot`, then calculates the distribution of Cl-bearing species. Based on mass balance of Na atoms, the molality of NaCl is then used to recalculate the molality of Na^+, followed by ionic strength. To find a solution, the function iterates until the change of molality of Na^+ and ionic strength are both less than `m_tot` / 100.

At very high NaCl concentrations, which are beyond the applicability limits of the extended Debye-Hückel model and therefore not recommended for normal use, the iterations tend to oscillate without converging. Setting `attenuate` to TRUE, which halves the amount of change in each step, may

help with convergence. If a solution is not found after 100 iterations, the function stops with an error.

If `pH` is NA (the default), then HCl is not included in the calculation and its molality in the output is also assigned NA. Note that only a single value is accepted for `m_tot`, but the other numeric arguments can have length > 1, allowing multiple combinations `T`, `P`, and `pH` in a single function call. However, due to limitations in `affinity`, only one of `T` and `P` can have length > 1.

Value

A list with components '`IS`' (ionic strength calculated from molalities of Na^+ and Cl^-), '`m_Cl`', '`m_Cl`', '`m_NaCl`', and '`m_HCl`' (molalities of Na^+, Cl^-, NaCl, and HCl).

Warning

It is important to keep in mind the ionic strength limits of the Debye-Hückel equation, but this function doesn't enforce them. Furthermore, metal-ligand complexing is not calculated by this function, so metal solubility and speciation calculations will be accurate only for relatively insoluble metals in NaCl-dominated solutions.

References

Shvarov, Y. and Bastrakov, E. (1999) HCh: A software package for geochemical equilibrium modelling. User's Guide. *Australian Geological Survey Organisation* **1999/25**. https://pid.geoscience.gov.au/dataset/ga/25473

See Also

This function is used in a few demos (`demo("contour")`, `demo("gold")`, `demo("minsol")`, `demo("sphalerite")`). `demo("yttrium")` uses the `pH` argument.

Examples

```
# Ionic strength calculated with HCh version 3.7 (Shvarov and Bastrakov, 1999)
# at 1000 bar, 100, 200, and 300 degress C, and 1 to 6 molal NaCl
m.HCh <- 1:6
IS.HCh <- list(`100` = c(0.992, 1.969, 2.926, 3.858, 4.758, 5.619),
               `300` = c(0.807, 1.499, 2.136, 2.739, 3.317, 3.875),
               `500` = c(0.311, 0.590, 0.861, 1.125, 1.385, 1.642))
# Total molality in the calculation with NaCl()
m_tot <- seq(1, 6, 0.5)
N <- length(m_tot)
# Where we'll put the calculated values
IS.calc <- data.frame(`100` = numeric(N), `300` = numeric(N), `500` = numeric(N))
# NaCl() is *not* vectorized over m_tot, so we use a loop here
for(i in 1:length(m_tot)) {
  NaCl.out <- NaCl(m_tot[i], c(100, 300, 500), P = 1000)
  IS.calc[i, ] <- NaCl.out$IS
}
# Plot ionic strength from HCh and NaCl() as points and lines
col <- c("black", "red", "orange")
plot(c(1,6), c(0,6), xlab = "NaCl (mol/kg)", ylab = axis.label("IS"), type = "n")
```

```
for(i in 1:3) {
  # NOTE: the differences are probably mostly due to different models
  # for the properties of NaCl(aq) (HCh: B.Ryhzenko model;
  # CHONSZ: revised HKF with parameters from Shock et al., 1997)
  points(m.HCh, IS.HCh[[i]], col = col[i])
  lines(m_tot, IS.calc[, i], col = col[i])
}
# Add legend and title
dprop <- describe.property(rep("T", 3), c(100, 300, 500))
legend("topleft", dprop, lty = 1, pch = 1, col = col)
title(main="H2O + NaCl; HCh (points) and 'NaCl()' (lines)")
```

`nonideal` *Activity Coefficients of Aqueous Species*

Description

Calculate activity coefficients and adjusted molal properties of aqueous species.

Usage

```
nonideal(species, speciesprops, IS, T, P, A_DH, B_DH,
         m_star = NULL, method = thermo()$opt$nonideal)
bgamma(TC, P, showsplines = "")
```

Arguments

`species`	name of method to use, or names or indices of species for which to calculate nonideal properties
`speciesprops`	list of dataframes of species properties
`IS`	numeric, ionic strength(s) used in nonideal calculations, mol kg^{-1}
`T`	numeric, temperature (K)
`P`	numeric, pressure (bar); required for B-dot or b_gamma equation
`A_DH`	numeric, A Debye-Huckel coefficient; required for B-dot or b_gamma equation
`B_DH`	numeric, B Debye-Huckel coefficient; required for B-dot or b_gamma equation
`m_star`	numeric, total molality of all dissolved species
`method`	character, '`Bdot`', '`Bdot0`', '`bgamma`', '`bgamma0`', or '`Alberty`'
`TC`	numeric, temperature (°C)
`showsplines`	character, show isobaric ('`T`') or isothermal ('`P`') splines

Details

`nonideal` calculates activity coefficients and adjusted thermodynamic properties for charged and neutral aqueous species. At the user level, the main use of this function is to set the method for activity coefficient calculations that gets used by other functions in CHNOSZ. See the "Charged Species" section for a description of the available methods. Activity coefficient calculations are activated by setting the `IS` argument of `subcrt` or `affinity`. Those functions then call `nonideal` with all the arguments needed to perform the calculations.

Charged Species

The default is to not apply calculations for the proton (H^+) and electron (e^-); this makes sense if you are setting the pH, i.e. activity of H^+, to some value. To apply the calculations to H+ and/or e-, change `thermo()$opt$ideal.H` or `ideal.e` to FALSE (see examples).

For the '`Alberty`' method, the values of `IS` are combined with Alberty's (2003) equation 3.6-1 (extended Debye-Hückel equation with an empirical term valid up to 0.25 M ionic strength) and its derivatives (Alberty, 2001), to calculate adjusted molal properties at the specified ionic strength(s) and temperature(s). The calculations use the equation for the Debye-Hückel constant given by Clarke and Glew, 1980, which is valid between 0 and 150 °C at saturated water vapor pressure (P_{SAT}).

For the '`Bdot`' method (the default), the "B-dot" form of the extended Debye-Hückel equation is used. This equation is valid at ionic strengths up to approximately 3 mol / kg (Hörbrand et al., 2018). The distance of closest approach for different ions (the "ion size parameter") is taken from `thermo()$Bdot_acirc`; any species not listed in this file is assigned a value of 4.5 Å. The extended term parameter for NaCl-dominated solutions, known as "B-dot", is calculated as a function only of temperature (Helgeson, 1969). To set the extended term parameter to zero, use the '`Bdot0`' method.

For the '`bgamma`' method, the "b_gamma" equation is used. The distance of closest approach is set to a constant (3.72e-8 cm) (e.g., Manning et al., 2013). The extended term parameter is calculated by calling the `bgamma` function. Alternatively, set the extended term parameter to zero with '`bgamma0`'.

Neutral Species

For neutral species, the Setchénow equation is used, as described in Shvarov and Bastrakov, 1999. If `thermo()$opt$Setchenow` is '`bgamma0`' (the default), the extended term parameter is set to zero and the only non-zero term is the mole fraction to molality conversion factor (using the value of `m_star`). If `thermo()$opt$Setchenow` is '`bgamma`', the extended term paramter is taken from the setting for the charged species (which can be either '`Bdot`' or '`bgamma`'). Set `thermo()$opt$Setchenow` to any other value to disable the calculations for neutral species.

Additional Details

This information, about the arguments and return values used to perform the calculations, is not normally needed by the user (but the usage is shown in the last example).

For `nonideal`, the species can be identified by name or species index in `species`. `speciesprops` is a list of dataframes containing the input standard molal properties; normally, at least one column is '`G`' for Gibbs energy. The function calculates the *adjusted* properties for given ionic strength (`IS`); they are equal to the *standard* values only at IS=0. The adjusted molal properties that can be calculated include '`G`', and (currently only with the Alberty method) '`H`', '`S`' and '`Cp`'; values of any columns with other names are left untouched. The lengths of `IS` and `T` supplied in the arguments should be equal to the number of rows of each dataframe in `speciesprops`, or length one to use single values throughout.

In addition to `IS` and `T`, the '`Bdot`' and '`bgamma`' methods depend on values of `P`, `A_DH`, `B_DH`, and `m_star` given in the arguments. `m_star`, the total molality of all dissolved species, is used to compute the mole fraction to molality conversion factor. If `m_star` is NULL, it is taken to be equal to `IS`, which is an underestimate. For these methods, '`G`' is currently the only adjusted

molal property that is calculated (but this can be used by `subcrt` to calculate adjusted equilibrium constants).

The return value is the same as the input in `speciesprops`, except the input standard thermodynamic properties (at IS=0) are replaced by adjusted properties (at higher IS). For all affected species, a column named `loggam` (common (base-10) logarithm of gamma, the activity coefficient) is appended to the output dataframe of species properties.

`bgamma` calculates the extended term parameter (written as b_gamma; Helgeson et al., 1981) for activity coefficients in NaCl-dominated solutions at high temperature and pressure. Data at P_{SAT} and 0.5 to 5 kb are taken from Helgeson (1969, Table 2 and Figure 3) and Helgeson et al. (1981, Table 27) and extrapolated values at 10 to 30 kb from Manning et al. (2013, Figure 11). Furthermore, the 10 to 30 kb data were used to generate super-extrapolated values at 40, 50, and 60 kb, which may be encountered using the `water.DEW` calculations. If all `P` correspond to one of the isobaric conditions, the values of `Bdot` at `T` are calculated by spline fits to the isobaric data. Otherwise, particular (dependent on the `T`) isobaric spline fits are themselves used to construct isothermal splines for the given values of `T`; the isothermal splines are then used to generate the values of `Bdot` for the given `P`. To see the splines, set `showsplines` to '`T`' to make the first set (isobaric splines) along with the data points, or '`P`' for examples of isothermal splines at even temperature intervals (here, the symbols are not data, but values generated from the isobaric splines). This is a basic method of interpolating the data without adding any external dependencies.

References

Alberty, R. A. (2001) Effect of temperature on standard transformed Gibbs energies of formation of reactants at specified pH and ionic strength and apparent equilibrium constants of biochemical reactions. *J. Phys. Chem. B* **105**, 7865–7870. doi:10.1021/jp011308v

Alberty, R. A. (2003) *Thermodynamics of Biochemical Reactions*, John Wiley & Sons, Hoboken, New Jersey, 397 p. https://www.worldcat.org/oclc/51242181

Clarke, E. C. W. and Glew, D. N. (1980) Evaluation of Debye-Hückel limiting slopes for water between 0 and 150 °C. *J. Chem. Soc. Faraday Trans.* **76**, 1911–1916. doi:10.1039/f19807601911

Helgeson, H. C. (1969) Thermodynamics of hydrothermal systems at elevated temperatures and pressures. *Am. J. Sci.* **267**, 729–804. doi:10.2475/ajs.267.7.729

Helgeson, H. C., Kirkham, D. H. and Flowers, G. C. (1981) Theoretical prediction of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures. IV. Calculation of activity coefficients, osmotic coefficients, and apparent molal and standard and relative partial molal properties to 600°C and 5 Kb. *Am. J. Sci.* **281**, 1249–1516. doi:10.2475/ajs.281.10.1249

Hörbrand, T., Baumann, T. and Moog, H. C. (2018) Validation of hydrogeochemical databases for problems in deep geothermal energy. *Geotherm. Energy* **6**, 20. doi:10.1186/s4051701801063

Manning, C. E. (2013) Thermodynamic modeling of fluid-rock interaction at mid-crustal to upper-mantle conditions. *Rev. Mineral. Geochem.* **76**, 135–164. doi:10.2138/rmg.2013.76.5

Manning, C. E., Shock, E. L. and Sverjensky, D. A. (2013) The chemistry of carbon in aqueous fluids at crustal and upper-mantle conditions: Experimental and theoretical constraints. *Rev. Mineral. Geochem.* **75**, 109–148. doi:10.2138/rmg.2013.75.5

Shvarov, Y. and Bastrakov, E. (1999) HCh: A software package for geochemical equilibrium modelling. User's Guide. *Australian Geological Survey Organisation* **1999/25**. https://pid.geoscience.gov.au/dataset/ga/25473

Examples

```
## Each of the available methods
nonideal("Alberty")
nonideal("bgamma0")
nonideal("bgamma")
nonideal("Bdot0")
nonideal("Bdot") # the default

## What's the activity coefficient of Na+ at
## 25 degC and 1 bar and an ionic strength of 0.7?
sres <- subcrt("Na+", T = 25, IS = 0.7)
# Exponentiate to convert log10(gamma) to gamma
10^sres$out[[1]]$loggam
# Now use a different method
nonideal("bgamma")
sres <- subcrt("Na+", T = 25, IS = 0.7)
10^sres$out[[1]]$loggam

## What are activity coefficients of -3, -2, -1, 0, +1, +2, +3 charged species
## as a function of ionic strength and temperature?
# First choose the method
nonideal("Bdot")
# Define the ionic strength and temperature increments
IS <- c(0.001, 0.01, 0.1, 0.7)
T <- seq(0, 100, 25)
# Use species with charges -3, -2, -1, 0, +1, +2, +3
species <- c("PO4-3", "HPO4-2", "H2PO4-", "H3PO4", "Na+", "Ca+2", "Al+3")
# Initialize empty output table for T (rows) and charge (columns)
gamtab <- matrix(nrow = length(T), ncol = length(species))
rownames(gamtab) <- T
colnames(gamtab) <- -3:3
# Make a list of tables to hold the activity coefficients, one for each IS
gamma <- rep(list(gamtab), length(IS))
names(gamma) <- IS
# Loop over the values of ionic strength
for(i in seq_along(IS)) {
  # Calculate properties of species, including logarithm of activity coefficient
  sres <- subcrt(species, T = T, IS = IS[i])
  # Exponentiate to convert log10(gamma) to gamma, and put the values into the tables
  for(j in seq_along(species)) gamma[[i]][, j] <- 10^sres$out[[j]]$loggam
}
# Print the output and make a plot
print(gamma)
matplot(T, gamma$`0.001`, type = "l")
title(main = "activity coefficients of -3, -2, -1, 0, +1, +2, +3 charged species")

## Alberty, 2003 p. 16 Table 1.3: adjusted pKa of acetic acid
## We use the 'IS' argument in subcrt() to calculate adjusted thermodynamic properties
# Set ideal.H to FALSE to calculate activity coefficients for the proton
# (needed for replication of the values in Alberty's book)
nonideal("Alberty")
thermo("opt$ideal.H" = FALSE)
```

```
sres <- subcrt(c("acetate", "H+", "acetic acid"), c(-1, -1, 1),
  IS=c(0, 0.1, 0.25), T=25, property="logK")
# We're within 0.01 of Alberty's pK values
Alberty_logK <- c(4.75, 4.54, 4.47)
# The maximum (absolute) pairwise difference between x and y
max(abs(Alberty_logK - sres$out$logK)) # 0.0072
# Reset option to default
thermo("opt$ideal.H" = TRUE)

## An example using IS with affinity():
## Speciation of phosphate as a function of ionic strength
opar <- par(mfrow = c(2, 1))
basis("CHNOPS+")
Ts <- c(25, 100)
species(c("PO4-3", "HPO4-2", "H2PO4-"))
for(T in Ts) {
  a <- affinity(IS = c(0, 0.14), T = T)
  e <- equilibrate(a)
  if(T==25) diagram(e, ylim = c(-3.0, -2.6), legend.x = NULL)
  else diagram(e, add = TRUE, names = FALSE, col = "red")
}
title(main="Non-ideality model for phosphate species")
dp <- describe.property(c("pH", "T", "T"), c(7, Ts))
legend("topright", lty = c(NA, 1, 1), col = c(NA, "black", "red"), legend = dp)
text(0.07, -2.76, expr.species("HPO4-2"))
text(0.07, -2.90, expr.species("H2PO4-"))
## Phosphate predominance f(IS,pH)
a <- affinity(IS = c(0, 0.14), pH = c(6, 13), T = Ts[1])
d <- diagram(a, fill = NULL)
a <- affinity(IS = c(0, 0.14), pH = c(6, 13), T = Ts[2])
d <- diagram(a, add = TRUE, names = FALSE, col = "red")
par(opar)

## Activity coefficients for monovalent ions at 700 degC, 10 kbar
# After Manning, 2013, Fig. 7
# Here we use the b_gamma equation
nonideal("bgamma")
IS <- c(0.001, 0.01, 0.1, 1, 2, 2.79)
# We're above 5000 bar, so need to use IAPWS-95 or DEW
oldwat <- water("DEW")
sres <- subcrt("Na+", T = 700, P = 10000, IS = IS)
water(oldwat)
# Compare the calculated activity coefficient to values from Manning's figure
gamma <- 10^sres$out[[1]]$loggam
Manning_gamma <- c(0.93, 0.82, 0.65, 0.76, 1.28, 2)
gamma - Manning_gamma

## Plot the data and splines used for calculating b_gamma
## (extended term parameter)
bgamma(showsplines = "T")
bgamma(showsplines = "P")

## A longer example, using nonideal() directly
```

```
# Alberty, 2003 p. 273-276: activity coefficient (gamma)
# as a function of ionic strength and temperature
nonideal("Alberty")
IS <- seq(0, 0.25, 0.005)
T <- c(0, 25, 40)
lty <- 1:3
species <- c("H2PO4-", "HADP-2", "HATP-3", "ATP-4")
col <- rainbow(4)
thermo.plot.new(xlim = range(IS), ylim = c(0, 1),
  xlab = axis.label("IS"), ylab = "gamma")
for(j in 1:3) {
  # Use subcrt to generate speciesprops
  speciesprops <- subcrt(species, T = rep(T[j], length(IS)))$out
  # Use nonideal to calculate loggamma; this also adjusts G, H, S, Cp,
  # but we don't use them here
  nonidealprops <- nonideal(species, speciesprops, IS = IS, T = convert(T[j], "K"))
  for(i in 1:4) lines(IS, 10^(nonidealprops[[i]]$loggam), lty=lty[j], col=col[i])
}
t1 <- "Activity coefficient (gamma) of -1,-2,-3,-4 charged species"
t2 <- quote("at 0, 25, and 40 "*degree*"C, after Alberty, 2003")
mtitle(as.expression(c(t1, t2)))
legend("topright", lty=c(NA, 1:3), bty="n",
  legend=c(as.expression(axis.label("T")), 0, 25, 40))
legend("top", lty=1, col=col, bty="n",
  legend = as.expression(lapply(species, expr.species)))

## Reset method to default
nonideal("Bdot")  # or reset()
```

`palply` *Conditional Parallel Processing*

Description

Use multiple processors for large calculations.

Usage

```
palply(varlist, X, FUN, ...)
```

Arguments

`...`	equivalent to the same argument in `parallel::parLapply`
`varlist`	character, names of variables to export using `parallel::clusterExport`
`X`	vector, argument for `lapply` or `parLapply`
`FUN`	function, argument for `lapply` or `parLapply`

Details

palply is a wrapper function to run parallel::parLapply if length of X > thermo()optparamin and multiple cores are available, otherwise it runs lapply. Note that parLapply is called with methods set to FALSE. If lots of package startup messages are created when running parallel::makeCluster (which is called by palply), it can probably be stopped by adding a test for interactive sessions around any library commands in the Rprofile.

See Also

read.fasta, count.aa, affinity, equil.boltzmann and equil.reaction for functions that use palply. Tests are in 'tests/test-util.program.R', and a "real world" example is in 'demo/density.R'.

---

protein.info *Summaries of Thermodynamic Properties of Proteins*

Description

Protein information, length, chemical formula, thermodynamic properties by group additivity, reaction coefficients of basis species, and metastable equilibrium example calculation.

Usage

```
pinfo(protein, organism=NULL, residue=FALSE, regexp=FALSE)
protein.length(protein, organism = NULL)
protein.formula(protein, organism = NULL, residue = FALSE)
protein.OBIGT(protein, organism = NULL, state=thermo()$opt$state)
protein.basis(protein, T = 25, normalize = FALSE)
protein.equil(protein, T=25, loga.protein = 0, digits = 4)
```

Arguments

protein	character, names of proteins; numeric, species index of proteins; data frame; amino acid composition of proteins
organism	character, names of organisms
residue	logical, return per-residue values (those of the proteins divided by their lengths)?
regexp	logical, find matches using regular expressions?
normalize	logical, return per-residue values (those of the proteins divided by their lengths)?
state	character, physical state
T	numeric, temperature in °C
loga.protein	numeric, decimal logarithms of reference activities of proteins
digits	integer, number of significant digits (see signif)

Details

For character protein, pinfo returns the rownumber(s) of thermo()$protein that match the protein names. The names can be supplied in the single protein argument (with an underscore, denoting protein_organism) or as pairs of proteins and organisms. NA is returned for any unmatched proteins, including those for which no organism is given or that do not have an underscore in protein.

Alternatively, if regexp is TRUE, the protein argument is used as a pattern (regular expression); rownumbers of all matches of thermo()$protein$protein to this pattern are returned. When using regexp, the organism can optionally be provided to return only those entries that also match thermo()$protein$organism.

For numeric protein, pinfo returns the corresponding row(s) of thermo()$protein. Set residue to TRUE to return the per-residue composition (i.e. amino acid composition of the protein divided by total number of residues).

For dataframe protein, pinfo returns it unchanged, except for possibly the per-residue calculation.

The following functions accept any specification of protein(s) described above for pinfo:

protein.length returns the lengths (number of amino acids) of the proteins.

protein.formula returns a stoichiometrix matrix representing the chemical formulas of the proteins that can be pased to e.g. mass or ZC. The amino acid compositions are multiplied by the output of group.formulas to generate the result.

protein.OBIGT calculates the thermodynamic properties and equations-of-state parameters for the completely nonionized proteins using group additivity with parameters taken from Dick et al., 2006 (aqueous proteins) and LaRowe and Dick, 2012 (crystalline proteins and revised aqueous methionine sidechain group). The return value is a data frame in the same format as thermo()$OBIGT. state indicates the physical state for the parameters used in the calculation ('aq' or 'cr').

The following functions also depend on an existing definition of the basis species:

protein.basis calculates the numbers of the basis species (i.e. opposite of the coefficients in the formation reactions) that can be combined to form the composition of each of the proteins. The basis species must be present in thermo()$basis, and if 'H+' is among the basis species, the ionization states of the proteins are included. The ionization state of the protein is calculated at the pH defined in thermo()$basis and at the temperature specified by the T argument. If normalize is TRUE, the coefficients on the basis species are divided by the lengths of the proteins.

protein.equil produces a series of messages showing step-by-step a calculation of the chemical activities of proteins in metastable equilibrium. For the first protein, it shows the standard Gibbs energies of the reaction to form the nonionized protein from the basis species and of the ionization reaction of the protein (if 'H+' is in the basis), then the standard Gibbs energy/RT of the reaction to form the (possibly ionized) protein per residue. The per-residue values of 'logQstar' and 'Astar/RT' are also shown for the first protein. Equilibrium calculations are then performed, only if more than one protein is specified. This calculation applies the Boltzmann distribution to the calculation of the equilibrium degrees of formation of the residue equivalents of the proteins, then converts them to activities of proteins taking account of loga.protein and protein length. If the protein argument is numeric (indicating rownumbers in thermo()$protein), the values of 'Astar/RT' are compared with the output of affinity, and those of the equilibrium degrees of

formation of the residues and the chemical activities of the proteins with the output of diagram. If the values in any of these tests are are not all.equal an error is produced indicating a bug.

References

Dick, J. M., LaRowe, D. E. and Helgeson, H. C. (2006) Temperature, pressure, and electrochemical constraints on protein speciation: Group additivity calculation of the standard molal thermodynamic properties of ionized unfolded proteins. *Biogeosciences* **3**, 311–336. doi:10.5194/bg33112006

LaRowe, D. E. and Dick, J. M. (2012) Calculation of the standard molal thermodynamic properties of crystalline peptides. *Geochim. Cosmochim. Acta* **80**, 70–91. doi:10.1016/j.gca.2011.11.041

Examples

```
# Search by name in thermo()$protein
# These are the same: ip1 == ip2
ip1 <- pinfo("LYSC_CHICK")
ip2 <- pinfo("LYSC", "CHICK")
# Two organisms with the same protein name
ip3 <- pinfo("MYG", c("HORSE", "PHYCA"))
# Their amino acid compositions
pinfo(ip3)
# Their thermodynamic properties by group additivity
protein.OBIGT(ip3)

# An unknown protein name gives NA
ip4 <- pinfo("MYGPHYCA")

## Example for chicken lysozyme C
# Index in thermo()$protein
ip <- pinfo("LYSC_CHICK")
# Amino acid composition
pinfo(ip)
# Protein length and chemical formula
protein.length(ip)
protein.formula(ip)
# Group additivity for thermodynamic properties and HKF equation-of-state
# parameters of non-ionized protein
protein.OBIGT(ip)
# Calculation of standard thermodynamic properties
# (subcrt uses the species name, not ip)
subcrt("LYSC_CHICK")
# NOTE: subcrt() only shows the properties of the non-ionized
# protein, but affinity() uses the properties of the ionized
# protein if the basis species have H+

## These are all the same
protein.formula("P53_PIG")
protein.formula(pinfo("P53_PIG"))
protein.formula(pinfo(pinfo("P53_PIG")))
```

`rank.affinity` *Average Ranks of Chemical Affinities*

Description

Affinity rankings for groups of species.

Usage

```
rank.affinity(aout, groups, percent = TRUE)
```

Arguments

`aout`	list, output of `affinity`
`groups`	named list of indices (integer or numeric) for species in each group
`percent`	return average rank percentage for each group

Details

The affinities for all species are `rank`ed, then the mean ranking for the species in each group is calculated. The mean rankings of groups are converted to a percentage, or returned as-is if `percent` is FALSE. Note that the calculations are applied to each set of conditions individually (i.e., each grid point in the affinity `affinity` calculation).

Value

The average rankings are inserted into the `values` element of `aout`, and the names of the groups are inserted into the `species` element. The result can be used by `diagram` to make line plots or predominance diagrams (the predominance fields correspond to the groups with highest average ranking of affinity).

Note

The reaction coefficients in the `species` element of the returned value of `aout` are not valid. Because balancing on a basis species (i.e., dividing by its reaction coefficient) would be incorrect, `diagram` enforces `balance = 1` so that that average rankings are used without further modification.

See Also

```
demo("rank.affinity")
```

Examples

```
# Compare Rubisco proteins from three domains
datfile <- system.file("extdata/cpetc/rubisco.csv", package = "CHNOSZ")
fastafile <- system.file("extdata/protein/rubisco.fasta", package = "CHNOSZ")
dat <- read.csv(datfile)
aa <- read.fasta(fastafile)
groups <- sapply(c("A", "B", "E"), "==", dat$domain, simplify = FALSE)
names(groups) <- c("Archaea", "Bacteria", "Eukaryota")
ip <- add.protein(aa, as.residue = TRUE)
basis("QEC")
aout <- affinity(O2 = c(-74, -66, 100), H2O = c(-4, 4, 100), iprotein = ip)
arank <- rank.affinity(aout, groups = groups)
nspecies <- sapply(groups, sum)
names <- paste0(names(groups), " (", nspecies, ")")
diagram(arank, fill = "terrain", font = 2, names = names, format.names = FALSE)
title("Average affinity ranking of Rubisco proteins")
```

---

retrieve *Retrieve Species by Element*

Description

Retrieve species in the database containing one or more chemical elements.

Usage

```
retrieve(elements = NULL, ligands = NULL, state = NULL,
  T = NULL, P = "Psat", add.charge = TRUE, hide.groups = TRUE)
```

Arguments

elements	character, combination of elements, or list, elements in a chemical system
ligands	character, elements present in any ligands
state	character, filter the result on these state(s).
T	numeric, temperature where ΔG° of species must be not NA
P	numeric, pressure where ΔG° of species must be not NA
add.charge	logical, add charge to the system?
hide.groups	logical, exclude groups from the result?

Details

This function retrieves the species in the thermodynamic database (see thermo) that have the indicated elements. A character value of elements is interpreted as a combination of one or more elements that must be present in each species. A list value of elements is used to specify a chemical system – the species must contain one or more of the indicated elements, but no other

elements. ligands, if present, gives the elements that may be present in any ligands; this can be used to retrieve all species in a system bearing the elements (usually a single metal).

The result includes charged species if add.charge is TRUE (the default) or the user supplies the "element" of charge ('Z'). Results can be filtered on physical state by setting the state argument. Groups used in group-additivity calculations, which have names with square brackets (e.g. [-CH2-]), are excluded unless hide.groups is FALSE. A special argument value 'all' can be used to retrieve all species in the thermodynamic database, including filtering on state and hiding of the groups.

The return value is a named integer vector giving the species index (i.e. rownumber(s) of thermo()$OBIGT) with names corresponding to the chemical formulas of the species. If the electron is in the result, its name ('e-') is used instead of its chemical formula ('(Z-1)'). An empty (length 0) integer value is returned if no elements are specified or no species are retrieved.

Set T (and optionally P) to require that species have non-NA values of ΔG° at this temperature and pressure. These values are passed to subcrt and therefore have the units set in T.units and P.units.

To speed up operation, the function uses a precalculated stoichiometric matrix for the default database, which is loaded with the package (see thermo). If the function detects a change to any chemical formulas in the database, it updates the stoichiometric matrix using i2A.

See Also

info for basic database searches; **anintro.html** for a diagram made with retrieved species in the Mn-O-H system.

Examples

```
# Species index of Ti-bearing minerals
retrieve("Ti")
# Thermodynamic data for those minerals
info(retrieve("Ti"))

# All species that have Au
retrieve("Au")
# All species that have both Au and Cl
retrieve(c("Au", "Cl"))
# Au-Cl system: species that have Au and/or Cl,
# including charged species, but no other elements
retrieve(list("Au", "Cl"))
# All Au-bearing species in the Au-Cl system
retrieve("Au", "Cl")
# All uncharged Au-bearing species in the Au-Cl system
retrieve("Au", "Cl", add.charge = FALSE)

# Minerals in the system SiO2-MgO-CaO-CO2
retrieve(list("Si", "Mg", "Ca", "C", "O"), state = "cr")
```

`solubility` *Equilibrium Chemical Activities of Species*

Description

Calculate chemical activities of aqueous species in equilibrium with a mineral or gas.

Usage

```
solubility(iaq, ..., in.terms.of = NULL, dissociate = FALSE, find.IS = FALSE)
```

Arguments

`iaq`	character (names) or numeric (species indices) of aqueous species
`...`	arguments for `affinity` or `mosaic` (i.e. plotting variables)
`in.terms.of`	character, express the total solubility in terms of moles of this species
`dissociate`	logical, does the mineral undergo a dissociation reaction?
`find.IS`	logical, find the equilibrium ionic strength by iteration?

Details

`solubility` calculates the activities of aqueous species in equilibrium with one or more minerals or gases. The minerals or gases should be loaded as the formed `species`, and the aqueous species (including ions and/or neutral complexes) that can be produced by dissolution should be listed in the `iaq` argument. The definitions of plotting variables should be provided in `...`, which are passed as arguments to `affinity`, or to `mosaic` if the first one is named `bases`.

It must be possible to obtain a valid set of basis species by substituting each of the minerals or gases in the first position of the current `basis` defintion, and all of the aqueous species must include that basis species in their formation reactions. (This essentially means that all minerals, gases and aqueous species must share a common element, which is what the reactions are balanced on.)

For a single mineral or gas, the output of `solubility` can be used by `diagram` with `type = NULL` (the default) to plot the activities of the aqueous species or with `type = "loga.balance"` to plot the sum of activities of aqueous species, which corresponds to the solubility of the mineral or gas. This value corresponds to the total extent of dissolution of the mineral or gas; `in.terms.of` can be used to express this value in terms of another species or element. For example, for dissolution of gaseous S_2, `in.terms.of = "S"` gives the total amount of S in solution, which is twice the amount of S_2 dissolved. Likewise, the solubility of corundum (Al_2O_3) can be expressed in terms of the moles of Al^{+3} in solution (see the vignette **anintro.html**).

For multiple minerals, the function calculates the solubilities for each of the minerals separately; these are stored in the `loga.equil` element of the output. The overall *minimum* solubility among all the minerals at each point is stored in `loga.balance`. This corresponds to the total activity of dissolved species in equilibrium with the most stable mineral. In contrast to the situation for a single mineral or gas, `diagram` by default plots `loga.balance`; `type = "loga.equil"` can be used to plot the solubilities for the individual minerals or gases.

Backward Compatibility

For compatibility with previous versions of the function, the `iaq` argument can be the output of `affinity` or `mosaic` for aqueous species. The examples for ionic strength and dissociation reactions were designed for this calling style.

In this case the (single) mineral or gas being dissolved is taken from the current `basis` species. Usually, the basis species should be set up so that the first basis species represents the substance being dissolved (a mineral such as $CaCO_3$ or gas such as CO_2). This is treated as the conserved basis species, so it must be present in all of the formation reactions of the aqueous species.

The `species` should be defined to represent one set of aqueous species (including ions and/or neutral complexes) formed in solution, all involving the conserved basis species. For a dissociation reaction, the second basis species should be used to represent the counterion (cation or anion). Other variables (pH, ionic strength, activities of other basis species) should be defined in the call to `affinity` to make `iaq`.

Dissociation Reactions

The function perfoms some additional steps to calculate the solubility of something that dissociates (not just dissolves). For example, the dissolution of calcite ($CaCO_3$), involves the release of both calcium ions and different forms of carbonate in solution, depending on the pH. The equilibrium calculation must take account of the *total* activity of the shared ion (Ca^{+2}), which is unknown at the start of the calculation. The solution is found by recalculating the affinities, essentially working backward from the assumption that the dissociation didn't occur. The resulting activities correspond to equilibrium considering the system-wide activity of Ca^{+2}.

A *not recommended* alternative is to set `dissociate` to a numeric value corresponding to the number of dissociated species (i.e. 2 for a 1:1 electrolyte). This setting indicates to calculate activities on a per-reaction basis, where each reaction has its own (independent) activity of Ca^{+2}. That does not give a complete equilibrium in the system, but may be required to reproduce some published diagrams (see comment in the calcite example of `demo("solubility")`).

Ionic Strength

Set `find.IS` to TRUE to determine the final ionic strength due to dissolution of a substance in pure water. This works by calculating the ionic strength from the amounts of aqueous species formed, then re-running `affinity` with the calculated `IS` value. Note that for dissociation reactions, the ionic strength is calculated from both the ions present in the species definition and the counter ion, which should be the second basis species. The calculation is iterated until the ionic strength deviation at every point is lower than a preset tolerance (1e-4). Alternatively, speciation of counterions (e.g. ionized forms of carbonate or sulfate) can also be performed by using the `mosaic` function instead of `affinity`; this is used in the second example below.

Warning

This function has not been tested for systems that may form dimers or higher-order complexes (such as $Au_2S_2{}^{2-}$). Except for relatively simple systems, even after careful refinement, the results from CHNOSZ, which considers chemical activities as the independent variables, will not match the results from speciation-solubility (or Gibbs energy minimization) codes, where the system is defined by its bulk composition.

See Also

retrieve provides a way to list all of the aqueous species in the database that have the specified elements.

demo("solubility") shows solubilities of CO_2 and calcite calculated as a function of pH and T; note that for calcite, the dissociate argument is set to TRUE. demo("gold") shows solubility calculations for Au in aqueous solutions with hydroxide, chloride, and hydrosulfide complexes.

Solubility calculations for multiple minerals are used for generating isosolubility (aka. equisolubility) lines in demo("Pourbaix") and demo("minsol"). The latter demo combines the calculation of solubilities with a mosaic calculation to account for the speciation of aqueous sulfur.

Whereas solubility yields a stable equilibrium condition (the affinities of formation reactions of aqueous species are zero), equilibrate generates metastable equilibrium (the affinities of formation reactions are equal to each other, but not necessarily zero).

Examples

```
## EXAMPLE 1

# Calculate solubility of a single substance:
# Gaseous SO2 with a given fugacity
# Define basis species (any S-bearing basis species should be first)
basis(c("sulfur", "oxygen", "H2O", "H+"))
basis("pH", 6)
# Load the substances (minerals or gases) to be dissolved
species("sulfur dioxide", -20)
# List the formed aqueous species
# We can use retrieve() to identify all the possible aqueous species
iaq <- retrieve("S", c("O", "H"), "aq")
# Place arguments for affinity() after the first argument of solubility()
s1 <- solubility(iaq, O2 = c(-56, -46), T = 125, in.terms.of = "S")

# Calculate overall solubility for multiple substances:
# Gaseous S2 and SO2 with a given fugacity
basis(c("sulfur", "oxygen", "H2O", "H+"))
basis("pH", 6)
species(c("S2", "sulfur dioxide"), -20)
s2 <- solubility(iaq, O2 = c(-56, -46), T = 125, in.terms.of = "S")

# Make expressions for legends
S_ <- expr.species("SO2", "gas", -20, TRUE)
pH_ <- quote(pH == 6)
T_ <- lT(125)
lexpr <- lex(S_, pH_, T_)

# Make diagrams from the results of solubility calculations
layout(matrix(c(1, 3, 2, 3), nrow = 2))
# Logarithm of activity of aqueous species in equilibrium with SO2(gas)
diagram(s1, ylim = c(-15, 0))
```

```
diagram(s1, type = "loga.balance", col = 3, lwd = 3, add = TRUE)
legend("topright", legend = lexpr, bty = "n")
# Logarithm of concentration (parts per million) of aqueous species
sppm <- convert(s1, "logppm")
diagram(sppm, ylim = c(-10, 5))
diagram(sppm, type = "loga.balance", col = 3, lwd = 3, add = TRUE)
legend("topright", legend = lexpr, bty = "n")
par(xpd = NA)
text(-58, 6.5, paste("Solubility of gaseous SO2 (green line) is",
  "sum of concentrations of aqueous species"), cex = 1.2, font = 2)
par(xpd = FALSE)

# Show overall (minimum) solubility of multiple gases
diagram(s2, col = 4, lwd = 3)
# Show solubilities of individual gases
names <- info(species()$ispecies)$formula
diagram(s2, type = "loga.equil", names = names, add = TRUE)
title("Minimum solubility (blue line) corresponds to the most stable gas")
layout(matrix(1))

## EXAMPLE 2

## Two ways to calculate pH-dependent solubility of calcite
## with ionic strength determination
## Method 1: CO2 and carbonate species as formed species
basis(c("CO2", "Ca+2", "H2O", "O2", "H+"))
species("calcite")
iaq <- info(c("CO2", "HCO3-", "CO3-2"))
# Ionic strength calculations don't converge below around pH = 3
sa0 <- solubility(iaq, pH = c(4, 14), dissociate = TRUE)
saI <- solubility(iaq, pH = c(4, 14), dissociate = TRUE, find.IS = TRUE)
## Method 2: CO2 and carbonate species as basis species
basis(c("Ca+2", "CO2", "H2O", "O2", "H+"))
species("calcite")
iaq <- info("Ca+2")
bases <- c("CO2", "HCO3-", "CO3-2")
sm0 <- solubility(iaq, bases = bases, pH = c(4, 14), dissociate = TRUE)
smI <- solubility(iaq, bases = bases, pH = c(4, 14), dissociate = TRUE, find.IS = TRUE)
## Plot the results
plot(0, 0, xlab="pH", ylab="solubility, log mol", xlim = c(4, 14), ylim = c(-5, 2))
# Method 1 with/without ionic strength
lines(saI$vals[[1]], saI$loga.balance, lwd = 5, col = "lightblue")
lines(sa0$vals[[1]], sa0$loga.balance, lwd = 5, col = "pink")
# Method 2 with/without ionic strength
lines(smI$vals[[1]], smI$loga.balance, lty = 2)
lines(sm0$vals[[1]], sm0$loga.balance, lty = 2)
legend("topright", c("I = 0", "I = calculated", "mosaic method"),
       col = c("pink", "lightblue", "black"), lwd = c(5, 5, 1), lty = c(1, 1, 2))
title(main = "Solubility of calcite: Ionic strength and mosaic method")
# The two methods give nearly equivalent results
stopifnot(all.equal(sa0$loga.balance, sm0$loga.balance))
stopifnot(all.equal(saI$loga.balance, smI$loga.balance, tolerance = 0.003))
## NOTE: the second method (using mosaic) is slower, but is
```

```
## more flexible; e.g. complexes with Ca+2 could be included
```

`species` *Species of Interest*

Description

Define the species of interest in a system; modify their physical states and logarithms of activities.

Usage

```
species(species = NULL, state = NULL, delete = FALSE, add = FALSE,
  index.return = FALSE)
```

Arguments

`species`	character, names or formulas of species to add to the species definition; numeric, rownumbers of species to modify or delete
`state`	character, physical states; numeric, logarithms of activities or fugacities
`delete`	logical, delete the species identified by numeric values of `species` (or all species if that argument is missing)?
`add`	logical, delete a previous species definition instead of adding to it?
`index.return`	logical, return the affected rownumbers of `thermo()$species` instead of its contents?

Details

After defining the `basis` species of your system you can use `species` to identify the species of interest. A species is uniquely identified by a combination of a `name` and `state`, which are columns of the thermodynamic database in `thermo()$OBIGT`. For each match of `species` to the name, formula, or abbreviation of a species, and of `state` to the state ('`aq`', '`cr`', '`gas`', '`liq`'), the species is added to the current species definition in `thermo()$species`.

The `state` argument can be omitted, in which case the first matching species in any state is added (in many cases, this means the aqueous species). If there are multiple matches for a species, the one that is in the state given by `thermo()$opt$state` is chosen, otherwise the matching (or n'th matching duplicate) species is used. Note that the `state`s of species representing phases of minerals that undergo polymorphic transitions are coded as '`cr`' (lowest-T phase), '`cr2`', '`cr3`', `...` (phases with increasing temperature). If `state` is '`cr`' when one of these minerals is matched, all the polymorphs are added.

The data frame in `thermo()$species` holds the species names and indices as well as the stoichiometric reaction coefficients for the formation reaction from the basis species and the logarithms of activities or fugacities that are used by `affinity`. The default values for logarithms of activities are -3 for aqueous ('`aq`') species and 0 for others.

To modify the logarithms of activities of species (logarithms of fugacities for gases) provide one or more numeric values of `species` referring to the rownumbers of the species dataframe, or

species NULL, to modify all currently defined species. The values in state, if numeric, are interpreted as the logarithms of activities, or if character are interpreted as new states for the species. If species is numeric and delete is TRUE, these species are deleted from the dataframe; if the only argument is delete and it is TRUE, all the species are removed.

By default, when identifying new species, any previous species definition is removed. Set add to TRUE to add species to an existing definition.

Value

With no arguments or when adding species, species returns the value of thermo()$species, unless index.return is TRUE, when the function returns the rownumbers of thermo()$species having the new species. With 'delete=TRUE', the value is the definition that existed prior the deletion; with 'delete=TRUE' and 'species' not NULL, the number of species remaining after the selected ones have been deleted, or NULL if no species remain.

See Also

Use info to search the thermodynamic database without adding species to the system. basis is a prerequisite for species.

Examples

```
# Set up the basis species
basis("CHNOS")
# Define some aqueous species
species(c("CO2", "NH3"))
# Add some gases
species(c("CO2", "NH3"), "gas", add = TRUE)
# Delete the aqueous species
species(1:2, delete = TRUE)
# Modify the "logact" value
# (log10 of activity for aqueous species;
#  log10 of fugacity for gases)
species(1:2, c(-2, -5))
# Change the state to aqueous
species(1:2, "aq")
# Load a new species definition (deletes the old one first)
species(c("glycine", "alanine"))
# Delete all the species
species(delete = TRUE)

# Changing the elements in the basis definition
# causes species to be deleted
basis(c("CaO", "CO2", "H2O", "SiO2", "MgO", "O2"))
species(c("dolomite", "quartz", "calcite", "forsterite"))
basis(c("CO2", "H2O", "O2"))
species()  # NULL
```

stack_mosaic *Stacked mosaic diagram*

Description

Create a stacked mosaic diagram, where the species formed in the first layer become the basis species for the species formed in the second layer. The species in each layer are usually minerals with different metals; any bimetallic species are added to the second layer.

Usage

```
stack_mosaic(bases, species1, species2, species12, names = NULL,
  col = list(4, 3, 6), col.names = list(4, 3, 6), fill = NULL,
  dx = list(0, 0, 0), dy = list(0, 0, 0), srt = list(0, 0, 0),
  lwd = list(1, 1, 1), lty = list(1, 1, 1), loga_aq = NULL,
  plot.it = TRUE, ...)
```

Arguments

bases	basis species to be changed for each layer (commonly S-bearing aqueous species)
species1	species (minerals and/or aqueous species) with metal 1
species2	species with metal 2
species12	bimetallic species
names	character, species names (or chemical formulas) for labeling fields
col	line color
col.names	text color
fill	field color
dx	label x-offset
dy	label y-offset
srt	label rotation
lwd	line width
lty	line type
loga_aq	numeric, activity of aqueous species
plot.it	make plots?
...	arguments for mosaic and affinity

Details

stack_mosaic creates a stacked mosaic diagram following steps that are described in detail in the vignette **multi-metal.html**. Briefly, the first layer of the diagram is made by speciating the species in bases across the diagram to form the first set of species in species1. Then, both bases *and* species1 (the stable species at each point on the diagram) are used to form the second set of species, including those in both species2 *and* species12.

Note that basis has aqueous S species in the examples provided, and species1 consists of minerals and/or aqueous species with a single metal (e.g. Fe). species2 has minerals and/or aqueous species with a second metal (e.g. Cu), and species12 has bimetallic minerals. For "mixed" diagrams (where species1 or species2 has both minerals and aqueous species), use loga_aq to set the logarithms of activities of aqueous species. Here, only a single value of loga_aq is needed, unlike in mosaic, where a value for each set of basis species is required.

The plot parameters col, col.names, fill, dx, dy, srt, lwd, and lty should be length-3 lists (not vectors). The values of elements 1–3 of the list are recycled to the number of species in species1, species2, and species12, respectively.

For fill, the default is to use no fill except for species12, where the fill color is taken from col.names with added transparency. The default definition of fill is list(NA, NA, add.alpha(col.names[3], "50")).

Value

A list of length two containing the output of each of the diagram calls use to make the diagram.

Warning

The bimetallic species in species12 are shown as part of the second layer, although their formation is sensitive to the presence of stable species in the first layer. It follows that changing the order of layers (i.e., swapping species1 and species2) can affect the depiction of mineral assemblages that have species12. It is likely that only one of the alternatives is thermodynamically correct, but currently there is no check to determine which one it is.

Examples

```
# Define temperature (degrees C), pressure (bar), pH and logfO2 ranges
T <- 200
P <- "Psat"
res <- 200
pH <- c(0, 14, res)
O2 <- c(-48, -33, res)

# Define system: Fe-Cu-O-S-Cl
# NOTE: the basis species must include the first species listed
# in each of bases, species1, and species2 below
basis(c("pyrite", "Cu", "Cl-", "H2S", "H2O", "oxygen", "H+"))
basis("H2S", -2)
# Calculate solution composition for 1 mol/kg NaCl
NaCl <- NaCl(T = T, P = P, m_tot = 1)
basis("Cl-", log10(NaCl$m_Cl))

# Define arguments for stack_mosaic: Speciate aqueous sulfur
bases <- c("H2S", "HS-", "HSO4-", "SO4-2")
# Calculate stabilities of Fe-bearing minerals first
species1 <- c("pyrite", "pyrrhotite", "magnetite", "hematite")
# Calculate stabilities of Cu-bearing and FeCu-bearing minerals second
species2 <- c("copper", "cuprite", "tenorite", "chalcocite", "covellite")
species12 <- c("chalcopyrite", "bornite")
```

```
# Use abbreviations for Fe-bearing minerals and formulas for Cu-bearing minerals
names1 <- c("Py", "Po", "Mag", "Hem")
names2 <- info(info(species2))$formula
names12 <- info(info(species12))$formula
names <- list(names1, names2, names12)
# Adjust x-position for one species (chalcocite, Cu2S)
dx <- list(c(0, 0, 0, 0), c(0, 0, 0, 1, 0), c(0, 0))
# Use thick dashed lines for the bimetallic species
lwd <- list(1, 1, 2)
lty <- list(1, 1, 2)

# Make the diagram
stack_mosaic(bases, species1, species2, species12, names = names,
  dx = dx, lwd = lwd, lty = lty,
  pH = pH, O2 = O2, T = T, P = P, IS = NaCl$IS)
# Add legend and title
lTP <- lex(lTP(T, P))
db <- describe.basis(c(3:4))
legend("topright", c(lTP, db), bg = "white")
title("Fe-Cu-S-O-H-Cl", font.main = 1)
```

`subcrt` *Properties of Species and Reactions*

Description

Calculate the standard molal thermodynamic properties of one or more species or a reaction between species as a function of temperature and pressure.

Usage

```
subcrt(species, coeff = 1, state = NULL,
  property = c("logK","G","H","S","V","Cp"),
  T = seq(273.15,623.15,25), P = "Psat", grid = NULL,
  convert = TRUE, exceed.Ttr = FALSE, exceed.rhomin = FALSE,
  logact = NULL, autobalance = TRUE, use.polymorphs = TRUE, IS = 0)
```

Arguments

`species`	character, name or formula of species, or numeric, rownumber of species in `thermo()$OBIGT`
`coeff`	numeric, reaction coefficients on species
`state`	character, state(s) of species
`property`	character, property(s) to calculate
`T`	numeric, temperature(s) of the calculation
`P`	numeric, pressure(s) of the calculation, or character, '`Psat`'
`grid`	character, type of `P`×`T` grid to produce (NULL, the default, means no gridding)

`exceed.Ttr` logical, calculate Gibbs energies of mineral phases and other species beyond their transition temperatures?

`exceed.rhomin` logical, return properties of species in the HKF model below 0.35 g cm^{-3}?

`logact` numeric, logarithms of activities of species in reaction

`convert` logical, are units of T, P, and energy settable by the user (default) (see `T.units`)?

`autobalance` logical, attempt to automatically balance reaction with basis species?

`use.polymorphs` logical, automatically identify available polymorphs in OBIGT and use the stable one at each value of `T`?

`IS` numeric, ionic strength(s) at which to calculate adjusted molal properties, mol kg^{-1}

Details

`subcrt` calculates the standard molal thermodynamic properties of species and reactions as a function of temperature and pressure. For each of the `species` (a formula or name), optionally identified in a given `state`, the standard molal thermodynamic properties and equations-of-state parameters are retrieved via `info` (except for H_2O liquid). The standard molal properties of the species are computed using the thermodynamic `model` given for each species (see `thermo`). This function also calculates the thermodynamic properties of *reactions* by summing those of the respective species. This functionality is inspired the SUPCRT92 package (Johnson et al., 1992).

`T` and `P` denote the temperature and pressure for the calculations. The only valid non-numeric value is '`Psat`' for `P`, which is the default (see `water`). For calculations below 273.16 K, `P` set to 1, as P_{SAT} is not defined at subzero (°C) temperatures. At temperatures above the critical point of water, `P` must be set to a numeric value; unless `exceed.rhomin` is TRUE, this should correspond to a fluid density $\geq$ 0.35 g cm^{-3}.

Argument `grid` if present can be one of `T` or `P` to perform the computation of a `T`×`P` or `P`×`T` grid. The `property`s to be calculated can be one or more of those shown below:

`rho`	Density of water	g cm^{-3}
`logK`	Logarithm of equilibrium constant	dimensionless
`G`	Gibbs energy	(cal | J) mol^{-1}
`H`	Enthalpy	(cal | J) mol^{-1}
`S`	Entropy	(cal | J) K^{-1} mol^{-1}
`V`	Volume	cm^3 mol^{-1}
`Cp`	Heat capacity	(cal | J) K^{-1} mol^{-1}
`E`	Exapansibility	cm^3 K^{-1}
`kT`	Isothermal compressibility	cm^3 bar^{-1}

If `convert` is `TRUE` (the default), the input values of `T` and `P` are interpreted to have the units given by `T.units` and `P.units` (default: °C and bar), and the output values of `G`, `H`, `S` and `Cp` are based on the units given in `E.units` (default: Joules). If `convert` is `FALSE`, the user units (`T.units`, `P.units`, and `E.units`) are ignored, and `T` and `P` are taken to be in Kelvin and bar, and the returned values of `G`, `H`, `S` and `Cp` are in Joules.

A chemical reaction is defined if coeff is given. In this mode the standard molal properties of species are summed according to the stoichiometric coefficients, where negative values denote reactants. An unbalanced reaction is signalled if the amount of any element on the reactant and product sides differs by more than 1e-7; in this case, subcrt prints the missing composition needed to balance the reaction and produces a warning but computes a result anyway. Alternatively, if autobalance is TRUE, the basis species of a system were previously defined, and all elements in the reaction are represented by the basis species, an unbalanced reaction given in the arguments to subcrt will be balanced automatically. The auto balancing doesn't change the reaction coefficients of any species in the reaction that are not among the basis species.

If logact is provided, the chemical affinities of reactions are calculated. logact indicates the logarithms of activities (fugacities for gases) of species in the reaction; if there are fewer values of logact than number of species those values are repeated as necessary. If the reaction was unbalanced to start, the logarithms of activities of any basis species added to the reaction are taken from the current definition of the basis species. Columns appended to the output are logQ for the log10 of the activity product of the reaction, and A for the chemical affinity, in the units set by E.units. Note that affinity provides related functionality but is geared toward the properties of formation reactions of species from the basis species and can be performed in more dimensions. Calculations of chemical affinity in subcrt can be performed for any reaction of interest; however, they are currently limited to constant values of the logarithms of activities of species in the reactions, and hence of logQ, across the computational range.

If IS is set to a single value other than zero, nonideal is used to calculate the adjusted properties (G, H, S and Cp) of charged aqueous species at the given ionic strength. To perform calculations at a single P and T and for multiple values of ionic strength, supply these values in IS. Calculations can also be performed on a P-IS, T-IS or P,T-IS grid. Values of logK of reactions calculated for IS not equal to zero are consistent with the adjusted Gibbs energies of the charged aqueous species.

If thermo()optvarP is TRUE, standard Gibbs energies of gases will be converted from a standard state at 1 bar (as used in SUPCRT) to a variable pressure standard state (see chapter 12 in Anderson and Crerar, 1993). This is useful for constructing e.g. boiling curves for organic compounds.

Value

A list of length two or three. If the properties of a reaction were calculated, the first element of the list (named 'reaction') contains a dataframe with the reaction parameters; the second element, named 'out', is a dataframe containing the calculated properties. Otherwise, the properties of species (not reactions) are returned: the first element, named 'species', contains a dataframe with the species identification; the second element, named 'out', is itself a list, each element of which is a dataframe of properties for a given species. If minerals with polymorphic transitions are present, a third element (a dataframe) in the list indicates for all such minerals the stable phase at each grid point.

Warning

Although SUPCRT92 prohibits calculations above 350 °C at P_{SAT} ("beyond range of applicability of aqueous species equations"), CHNOSZ does not impose this limitation, and allows calculations up to the critical temperature (373.917 °C) at P_{SAT}. Interpret calculations between 350 °C and the critical temperature at P_{SAT} at your own risk. The discontinuity in the value of $\log K$ at P_{SAT} that is apparent in demo("NaCl") demonstrates one unexpected result.

NAs are produced for calculations at 'Psat' when the temperature exceeds the critical temperature of H_2O. In addition, properties of species using the revised HKF equations are set to NA wherever the density of H_2O < 0.35 g/cm^3 (threshold just above the critical isochore; Johnson et al., 1992). Both of these situations produce warnings, which are stored in the 'warnings' element of the return value.

NAs are also output if the T, P conditions are otherwise beyond the capabilities of the water equations of state derived from SUPCRT92 (H2O92D.f), but the messages about this are produced by water.SUPCRT92 rather than subcrt.

Limitations

Note that E and kT can only be calculated for aqueous species and only if the option (thermo()optwater) for calculations of properties using water is set to IAPWS. On the other hand, if the water option is 'SUPCRT' (the default), E and kT can be calculated for water but not for aqueous species. (This is not an inherent limitation in either formulation, but it is just a matter of implementation.)

Note on polymorphic transitions

Minerals with polymorphic transitions (denoted in OBIGT by having states 'cr' (lowest-T phase), 'cr2', etc.) can be specified by name with 'cr' for the state or by using a numeric species index for the lowest-T polymorph. If use.polymorphs is TRUE, subcrt uses the transition temperatures calculated from those at P = 1 bar given in OBIGT together with functions of the entropies and volumes of transitions (see dPdTtr) to determine the stable polymorph at each grid point and uses the properties of that polymorph in the output. A polymorph column is added to the output to indicate the stable polymorph at each T-P condition. If exceed.Ttr is FALSE (the default), output values of Gibbs energy are assigned NA beyond the transition temperature of the highest-temperature polymorph. Set exceed.Ttr to TRUE to identify the stable polymorphs by comparing their extrapolated Gibbs energies instead of the tabulated transition temperatures. This is generally not advised, as extrapolated Gibbs energies may not reliably determine the stable polymorph at extreme temperatures.

References

Anderson, G. M. and Crerar, D. A. (1993) *Thermodynamics in Geochemistry: The Equilibrium Model*, Oxford University Press. https://www.worldcat.org/oclc/803272549

Johnson, J. W., Oelkers, E. H. and Helgeson, H. C. (1992) SUPCRT92: A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bar and 0 to 1000°C. *Comp. Geosci.* **18**, 899–947. doi:10.1016/0098-3004(92)90029Q

Helgeson, H. C., Owens, C. E., Knox, A. M. and Richard, L. (1998) Calculation of the standard molal thermodynamic properties of crystalline, liquid, and gas organic molecules at high temperatures and pressures. *Geochim. Cosmochim. Acta* **62**, 985–1081. doi:10.1016/S00167037(97)002196

LaRowe, D. E. and Helgeson, H. C. (2007) Quantifying the energetics of metabolic reactions in diverse biogeochemical systems: electron flow and ATP synthesis. *Geobiology* **5**, 153–168. doi:10.1111/j.14724669.2007.00099.x

Shock, E. L., Oelkers, E. H., Johnson, J. W., Sverjensky, D. A. and Helgeson, H. C. (1992) Calculation of the thermodynamic properties of aqueous species at high pressures and temperatures:

Effective electrostatic radii, dissociation constants and standard partial molal properties to 1000 °C and 5 kbar. *J. Chem. Soc. Faraday Trans.* **88**, 803–826. doi:10.1039/FT9928800803

See Also

`info` can be used to find species in the thermodynamic database. `makeup` is used by `subcrt` for parsing formulas to check mass balance of reactions. `demo("ORP")` and `nonideal` for examples using the `IS` argument.

Examples

```
## Properties of species
subcrt("water")
# Change temperature
subcrt("water", T = seq(0, 100, 20))
# Change temperature and pressure
T <- seq(500, 1000, 100)
P <- seq(5000, 10000, 1000)
subcrt("water", T = T, P = P)
# Temperature-pressure grid
subcrt("water", T = c(500, 1000), P = c(5000, 10000), grid = "P")

## Properties of reactions
subcrt(c("glucose", "ethanol", "CO2"), c(-1, 2, 2), T = 25)
# Use CO2(gas) (or just change "CO2" to "carbon dioxide")
subcrt(c("glucose", "ethanol", "CO2"), c(-1, 2, 2), c("aq", "aq", "gas"), T = 25)

## Automatically balance reactions
# First define the basis species
basis(c("CO2", "H2O", "NH3", "H2S", "O2"))
# Auto-balance adds the required amount of H2O and O2
subcrt(c("ethanol", "glucose"), c(-3, 1), T = 37)
# An example with H+
basis(c("H2O", "H2S", "O2", "H+"))
subcrt(c("HS-", "SO4-2"), c(-1, 1), T = 100)

## Mineral polymorphs
# Properties of the stable polymorph at each temperature
subcrt("pyrrhotite")
# Reactions automatically use stable polymorph
subcrt(c("pyrite", "pyrrhotite", "H2O", "H2S", "O2"), c(-1, 1, -1, 1, 0.5))
# Extrapolated properties of the lowest-T polymorph (metastable at higher temperatures)
subcrt(c("pyrrhotite"), use.polymorphs = FALSE, exceed.Ttr = TRUE)

## Messages about problems with the calculation
# Above the T, P limits for the H2O equations of state
subcrt("alanine", T = c(2250, 2251), P = c(30000, 30001), grid = "T")
# Psat is not defined above the critical point
# (suppressWarnings is used so that checks of examples don't raise warnings)
suppressWarnings(subcrt("alanine", T = seq(0, 5000, by = 1000)))

## Minerals with polymorphic transitions
```

```
# Compare calculated values of heat capacity of iron with
# values from Robie and Hemingway, 1995
T.units("K")
# We set pressure here otherwise subcrt uses Psat (saturation
# vapor pressure of H2O above 100 degrees C) which can't be
# calculated above the critical point of H2O (~647 K)
s <- subcrt("Fe", T=seq(300, 1800, 20), P=1)
plot(s$out[[1]]$T, s$out[[1]]$Cp, type="l",
  xlab=axis.label("T"), ylab=axis.label("Cp"))
# Add points from RH95
RH95 <- read.csv(system.file("extdata/cpetc/RH95.csv", package="CHNOSZ"))
points(RH95[,1], RH95[,2])
title(main=paste("Heat capacity of Fe(cr)\n",
  "(points - Robie and Hemingway, 1995)"))
# Reset the units to default values
T.units("C")

## Subzero (degrees C) calculations
# Uncomment the following to try IAPWS95 instead of SUPCRT92
#water("IAPWS95")
# The limit for H2O92D.f (from SUPCRT92) is currently -20 deg C
# but we go to -30 knowing properties will become NA
sb <- subcrt(c("H2O", "Na+"), T = seq(-30, 10), P = 1)$out
# Start plot with extra room on right
opar <- par(mar=c(5, 4, 4, 4))
# Plot Delta G
plot(sb$water$T, sb$water$G, ylim = c(-264000, -234000),
  xlab = axis.label("T"), ylab = axis.label("DG0"))
points(sb$`Na+`$T, sb$`Na+`$G, pch = 2)
# Add Cp
# change y-axis
par("usr" = c(par("usr")[1:2], -400, 100))
axis(4)
mtext(axis.label("Cp0"), side = 4, line = 3)
points(sb$water$T, sb$water$Cp, pch = 16)
points(sb$`Na+`$T, sb$`Na+`$Cp, pch = 17)
legend("topleft", c("H2O Cp", "H2O G", "Na+ Cp", "Na+ G"), pch = c(16, 1, 17, 2))
H2O <- expr.species("H2O")
Na <- expr.species("Na+")
degC <- expr.units("T")
title(main = substitute(H2O~and~Na~to~-20~degC, list(H2O = H2O, Na = Na, degC = degC)))
par(opar)

## Calculations using a variable-pressure standard state
thermo("opt$varP" = TRUE)
# Calculate the boiling point of octane at 2 and 20 bar
# We need exceed.Ttr = TRUE because the liquid is metastable
# at high temperatures (also, the gas is metastable at low
# temperatures, but that doesn't produce NA in the output)
sout2 <- subcrt(rep("octane", 2), c("liq", "gas"),
  c(-1, 1), T = seq(-50, 300, 0.1), P = 2, exceed.Ttr = TRUE)$out
sout20 <- subcrt(rep("octane", 2), c("liq", "gas"),
  c(-1, 1), T = seq(-50, 300, 0.1), P = 20, exceed.Ttr = TRUE)$out
```

```
# Find T with the Gibbs energy of reaction that is closest to zero
Tvap2 <- sout2$T[which.min(abs(sout2$G))]
Tvap20 <- sout20$T[which.min(abs(sout20$G))]
# Compare these with experimental values (Fig. 1 of Helgeson et al., 1998)
Tvap2.exp <- 156
Tvap20.exp <- 276
# Reset varP to FALSE (the default)
thermo("opt$varP" = FALSE)
```

`swap.basis` *Swap Basis Species*

Description

Swap the basis species defining a chemical system. One basis species is replaced by a new one with a different chemical formula.

Usage

```
swap.basis(species, species2, T = 25)
basis.elements(basis = thermo()$basis)
element.mu(basis = thermo()$basis, T = 25)
basis.logact(emu, basis = thermo()$basis, T = 25)
ibasis(species)
```

Arguments

`basis`	dataframe, a basis definition
`T`	numeric, temperature in Kelvin
`emu`	numeric, chemical potentials of elements
`species`	character, names or formulas of species, or numeric, indices of species
`species2`	character or numeric, a species to swap in to the basis definition

Details

`swap.basis` allows to change the basis definition by swapping out a basis species for a new one. Specify the names or formulas of the old and replacement basis species in the first argument. When the basis definition is changed, any species of interest that were present are deleted, unless the new basis definition has exactly the same elements as before. In that case, the species are kept; also, the activities of the new basis species are set in order to maintain the chemical potentials of the elements at `T` °C and 1 bar.

The other functions are supporting functions: `basis.elements` returns the stoichiometric matrix for the current basis definition, `element.mu` calculates the chemical potentials of elements corresponding to the activities of the basis species, `basis.logact` does the inverse operation, and `ibasis` returns the index in the basis set for a given species index (in `thermo$OBIGT`), name or formula.

See Also

basis, and mosaic

Examples

```
## Swapping basis species
# Start with a preset basis definition
b1 <- basis("CHNOS+")
# Swap H2(aq) for O2(gas)
b2 <- swap.basis("O2", "H2")
# Put oxygen back in
b3 <- swap.basis("H2", "oxygen")

# Interconversion of chemical potentials of elements and
# logarithms of activities of basis species at high temperature
basis("CHNOS+")
bl1 <- basis()$logact
emu <- element.mu(T = 100)
bl2 <- basis.logact(emu, T = 100)
# There's no difference
round(bl2 - bl1, 10)
```

---

taxonomy *Extract Data from NCBI Taxonomy Files*

Description

Read data from NCBI taxonomy files, traverse taxonomic ranks, get scientific names of taxonomic nodes.

Usage

```
getnodes(taxdir)
getrank(id, taxdir, nodes=NULL)
parent(id, taxdir, rank=NULL, nodes=NULL)
allparents(id, taxdir, nodes=NULL)
getnames(taxdir)
sciname(id, taxdir, names=NULL)
```

Arguments

taxdir	character, directory where the taxonomy files are kept.
id	numeric, taxonomic ID(s) of the nodes of interest.
nodes	dataframe, output from getnodes (optional).
rank	character, name of the taxonomic rank of interest.
names	dataframe, output from getnames (optional).

Details

These functions provide a convenient way to read data from NCBI taxonomy files (i.e., the contents of `taxdump.tar.gz`, which is available from https://ftp.ncbi.nih.gov/pub/taxonomy/).

The `taxdir` argument is used to specify the directory where the `nodes.dmp` and `names.dmp` files are located. `getnodes` and `getnames` read these files into data frames. `getrank` returns the rank (species, genus, etc) of the node with the given taxonomic `id`. `parent` returns the taxonomic ID of the next-lowest node below that specified by the `id` in the argument, unless `rank` is supplied, in which case the function descends the tree until a node with that rank is found. `allparents` returns all the taxonomic IDs of all nodes between that specified by `id` and the root of the tree, inclusive. `sciname` returns the scientific name of the node with the given `id`.

The `id` argument can be of length greater than 1 except for `allparents`. If `getrank`, `parent`, `allparents` or `sciname` need to be called repeatedly, the operation can be hastened by supplying the output of `getnodes` in the `nodes` argument and/or the output of `getnames` in the `names` argument.

Examples

```
## Get information about Homo sapiens from the
## packaged taxonomy files
taxdir <- system.file("extdata/taxonomy", package = "CHNOSZ")
# H. sapiens' taxonomic id
id1 <- 9606
# That is a species
getrank(id1, taxdir)
# The next step up the taxonomy
id2 <- parent(id1, taxdir)
print(id2)
# That is a genus
getrank(id2, taxdir)
# That genus is "Homo"
sciname(id2, taxdir)
# We can ask what phylum is it part of?
id3 <- parent(id1, taxdir, "phylum")
# Answer: "Chordata"
sciname(id3, taxdir)
# H. sapiens' complete taxonomy
id4 <- allparents(id1, taxdir)
sciname(id4, taxdir)

## The names of the organisms in the supplied taxonomy files
taxdir <- system.file("extdata/taxonomy", package = "CHNOSZ")
id5 <- c(83333, 4932, 9606, 186497, 243232)
sciname(id5, taxdir)
# These are not all species, though
# (those with "no rank" are something like strains,
# e.g. Escherichia coli K-12)
getrank(id5, taxdir)
# Find the species for each of these
id6 <- sapply(id5, function(x) parent(x, taxdir = taxdir, rank = "species"))
```

```
unique(getrank(id6, taxdir))  # "species"
# Note that the K-12 is dropped
sciname(id6, taxdir)

## The complete nodes.dmp and names.dmp files are quite large,
## so it helps to store them in memory when performing multiple queries
## (this doesn't have a noticeable speed-up for the small files in this example)
taxdir <- system.file("extdata/taxonomy", package = "CHNOSZ")
nodes <- getnodes(taxdir = taxdir)
# All of the node ids in this file
id7 <- nodes$id
# All of the non-leaf nodes
id8 <- unique(parent(id7, nodes = nodes))
names <- getnames(taxdir = taxdir)
sciname(id8, names = names)
```

`thermo` *Thermodynamic Database and System Settings*

Description

Run `reset()` to reset all of the data used in CHNOSZ to default values. This includes the computational settings, thermodynamic database, and system settings (chemical species).

The system settings are changed using `basis` and `species`. To clear the system settings (the default, i.e. no species loaded), run `basis("")`; to clear only the formed species, run `species(delete = TRUE)`

The thermodynamic database is changed using `add.OBIGT` and `mod.OBIGT`. To restore the default database without altering the species settings, run `OBIGT()`.

The computational settings are changed using `water`, `P.units`, `T.units`, `E.units`, and some other commands (e.g. `mod.buffer`).

All the data are stored in the `thermo` data object in an environment named `CHNOSZ`. `thermo()` is a convenience function to access or modify parts of this object, in particular some computational settings, for example, `thermo("opt$ideal.H" = FALSE)` (see `nonideal`).

The source data are provided with CHNOSZ as `*.csv` files in the `extdata/thermo` and `extdata/OBIGT` directories of the package. These files are used to build the `thermo` object, which is described below.

Usage

```
reset()
OBIGT(no.organics = FALSE)
thermo(...)
```

Arguments

`no.organics`	logical, load the database without data files for organic species (NOTE: CH_4 is listed as an "inorganic" species)?
`...`	list, one or more arguments whose names correspond to the setting to modify

Format

- thermo()$opt List of computational settings. Square brackets indicate default values. Note that the units of G.tol and Cp.tol depend on the E_units for each species in thermo()$OBIGT. Therefore, species with E_units of 'J' have a lower absolute tolerance for producing messages (because 4.184 J = 1 cal).

E.units	character	The user's units of energy (['J'] or 'cal') (see subcrt)
T.units	character	The user's units of temperature (['C'] or 'K')
P.units	character	The user's units of pressure (['bar'] or 'MPa')
state	character	The default physical state for searching species ['aq'] (see info)
water	character	Computational option for properties of water (['SUPCRT'] or 'IAPWS'; see water)
G.tol	numeric	Difference in G above which check.GHS produces a message (cal mol^{-1}) [100]
Cp.tol	numeric	Difference in Cp above which check.EOS produces a message (cal K^{-1} mol^{-1}) [1]
V.tol	numeric	Difference in V above which check.EOS produces a message (cm^3 mol^{-1}) [1]
varP	logical	Use variable-pressure standard state for gases? [FALSE] (see subcrt)
IAPWS.sat	character	State of water for saturation properties ['liquid'] (see util.water)
paramin	integer	Minimum number of calculations to launch parallel processes [1000] (see palply)
ideal.H	logical	Should nonideal ignore the proton? [TRUE]
ideal.e	logical	Should nonideal ignore the electron? [TRUE]
nonideal	character	Option for charged species in nonideal [Bdot]
Setchenow	character	Option for neutral species in nonideal [bgamma0]
Berman	character	User data file for mineral parameters in the Berman equations [NA]
maxcores	numeric	Maximum number of cores for parallel calculations with palply [2]
ionize.aa	numeric	Calculate properties of ionized proteins when H^+ is in basis species (see affinity) [TRUE]

- thermo()$element Dataframe containing the thermodynamic properties of elements taken from Cox et al., 1989, Wagman et al., 1982, and (for Am, Pu, Np, Cm) Thoenen et al., 2014. The standard molal entropy (S(Z)) at 25 °C and 1 bar for the "element" of charge (Z) was calculated from S(H2,g) + 2S(Z) = 2S(H+), where the standard molal entropies of H2,g and H+ were taken from Cox et al., 1989. The mass of Z is taken to be zero. Accessing this data frame using mass or entropy will select the first entry found for a given element; i.e., values from Wagman et al., 1982 will only be retrieved if the properties of the element are not found from Cox et al., 1989.

element	character	Symbol of element
state	character	Stable state of element at 25 °C and 1 bar
source	character	Source of data
mass	numeric	Mass of element (in natural isotopic distribution; referenced to a mass of 12 for ^{12}C)
s	numeric	Entropy of the compound of the element in its stable state at 25 °C and 1 bar (cal K^{-1} mol^{-1})
n	numeric	Number of atoms of the element in its stable compound at 25 °C and 1 bar

- thermo()$OBIGT
  This dataframe is a thermodynamic database of standard molal thermodynamic properties and equations of state parameters of species. "OrganoBioGeoTherm" is the name of a Windows GUI interface to SUPCRT92 that was produced in Harold C. Helgeson's Laboratory of The-

oretical Geochemistry and Biogeochemistry at the University of California, Berkeley. The OBIGT database was originally distributed with that program, and was the starting point for the database in CHNOSZ.

Note the following database conventions:

- The combination of `name` and `state` defines a species in `thermo()$OBIGT`. A species cannot be duplicated (this is checked when running `reset()`).
- English names of inorganic gases are used only for the gas state. The dissolved species is named with the chemical formula. Therefore, `info("oxygen")` refers to the gas, and `info("O2")` refers to the aqueous species.

Each entry is referenced to one or two literature sources listed in `thermo()$refs`. Use `thermo.refs` to look up the citation information for the references. See the vignette **OBIGT.html** for a complete description of the sources of data.

The identifying characteristics of species and their standard molal thermodynamic properties at 25 °C and 1 bar are located in the first 13 columns of `thermo()$OBIGT`:

`name`	character	Species name
`abbrv`	character	Species abbreviation
`formula`	character	Species formula
`state`	character	Physical state
`ref1`	character	Primary source
`ref2`	character	Secondary source
`date`	character	Date of data entry (ISO 8601 extended format)
`model`	character	Model for thermodynamic properties of the species
`E_units`	character	Units of energy: '`J`' for Joules or '`cal`' for calories
`G`	numeric	Standard molal Gibbs energy of formation from the elements (J|cal mol^{-1})
`H`	numeric	Standard molal enthalpy of formation from the elements (J|cal mol^{-1})
`S`	numeric	Standard molal entropy (J|cal mol^{-1} K^{-1})
`Cp`	numeric	Standard molal isobaric heat capacity (J|cal mol^{-1} K^{-1})
`V`	numeric	Standard molal volume (cm^3 mol^{-1})

`model` must be one of '`H2O`', '`HKF`', '`CGL`', '`Berman`', '`AD`', or '`DEW`'. '`H2O`' is reserved for liquid water, the properties of which are calculated using one of several available models (see `water`). Most aqueous species use '`HKF`' (the revised Helgeson-Kirkham-Flowers model). Properties of aqueous species with `model` set to '`AD`' are calculated using the Akinfiev-Diamond model, and those with '`DEW`' are calculated using the `DEW` model. Many minerals in the default database use the '`Berman`' model (see `Berman`). All other species use '`CGL`' (general crystalline, gas, liquid model). Properties of these species are calculated using a heat capacity function with up to six terms; the exponent on the final term can be defined in the database (see below).

The meanings of the remaining columns depend on the model for each species. The names of these columns are compounded from those of the parameters in the HKF equations of state and general heat capacity equation; for example, column 13 is named `a1.a`. Scaling of the values by integral powers of ten (i.e., orders of magnitude; OOM) for the HKF parameters (this also includes the DEW model) is based on the usual (but by no means universal) convention in the literature.

Columns 15-22 for aqueous species (parameters in the revised HKF equations of state). NOTE: Most older papers use units of calories for these parameters, so 'cal' is listed here; the actual units for each species are set in the E_units column.

a1	numeric	a_1 * 10 (cal mol^{-1} bar^{-1})
a2	numeric	a_2 * 10^{-2} (cal mol^{-1})
a3	numeric	a_3 (cal K mol^{-1} bar^{-1})
a4	numeric	a_4 * 10^{-4} (cal mol^{-1} K)
c1	numeric	c_1 (cal mol^{-1} K^{-1})
c2	numeric	c_2 * 10^{-4} (cal mol^{-1} K)
omega	numeric	ω * 10^{-5} (cal mol^{-1})
Z	numeric	Charge

Columns 15-22 for crystalline, gas and liquid species (C_P = a + bT + cT^{-2} + $dT^{-0.5}$ + eT^2 + fT^{lambda}). NOTE: As of CHNOSZ 2.0.0, OOM scaling for heat capacity coefficients has been removed, and new entries use units of Joules unless indicated by setting E_units to 'cal'.

a	numeric	a (J K^{-1} mol^{-1})
b	numeric	b (J K^{-2} mol^{-1})
c	numeric	c (J K mol^{-1})
d	numeric	d (J K$^{-0.5}$ mol^{-1})
e	numeric	e (J K^{-3} mol^{-1})
f	numeric	f (J K$^{-lambda-1}$ mol^{-1})
lambda	numeric	λ (exponent on the f term)
T	numeric	Positive value: Temperature (K) of polymorphic transition or phase stability limit
T	numeric	Negative value: Opposite of temperature (K) limit of C_P equation (see **FAQ.html** for details)

Columns 15-17 for aqueous species using the Akinfiev-Diamond model. Note that the c column is used to store the ξ parameter. Columns 18-22 are not used.

a	numeric	a (cm^3 g^{-1})
b	numeric	b (cm^3 K$^{0.5}$ g^{-1})
c	numeric	ξ
d	numeric	$XX1$ NA
e	numeric	$XX2$ NA
f	numeric	$XX3$ NA
lambda	numeric	$XX4$ NA
Z	numeric	Z NA

- thermo()$refs References for thermodynamic data.

key	character	Source key
author	character	Author(s)
year	character	Year
citation	character	Citation (journal title, volume, and article number or pages; or book or report title)

note	character	Short description of the compounds or species in this data source
URL	character	URL

- thermo()$buffers
  Dataframe which contains definitions of buffers of chemical activity. Each named buffer can be composed of one or more species, which may include any species in the thermodynamic database and/or any protein. The calculations provided by buffer do not take into account polymorphic transitions of minerals, so individual polymorphs of such minerals must be specified in the buffers.

name	character	Name of buffer
species	character	Name of species
state	character	Physical state of species
logact	numeric	Logarithm of activity (fugacity for gases)

- thermo()$protein Data frame of amino acid compositions of selected proteins. Most of the compositions were taken from the SWISS-PROT/UniProt online database (Boeckmann et al., 2003) and the protein and organism names usually follow the conventions adopted there. In some cases different isoforms of proteins are identified using modifications of the protein names; for example, 'MOD5.M' and MOD5.N proteins of 'YEAST' denote the mitochondrial and nuclear isoforms of this protein. See pinfo to search this data frame by protein name, and other functions to work with the amino acid compositions.

protein	character	Identification of protein
organism	character	Identification of organism
ref	character	Reference key for source of compositional data
abbrv	character	Abbreviation or other ID for protein
chains	numeric	Number of polypeptide chains in the protein
Ala...Tyr	numeric	Number of each amino acid in the protein

- thermo()$groups This is a dataframe with 22 columns for the amino acid sidechain, backbone and protein backbone groups ([Ala]..[Tyr],[AABB],[UPBB]) whose rows correspond to the elements C, H, N, O, S. It is used to quickly calculate the chemical formulas of proteins that are selected using the iprotein argument in affinity.
- thermo()$basis Initially NULL, reserved for a dataframe written by basis upon definition of the basis species. The number of rows of this dataframe is equal to the number of columns in "..." (one for each element).

...	numeric	One or more columns of stoichiometric coefficients of elements in the basis species
ispecies	numeric	Rownumber of basis species in thermo()$OBIGT
logact	numeric	Logarithm of activity or fugacity of basis species
state	character	Physical state of basis species

- thermo()$species Initially NULL, reserved for a dataframe generated by species to

define the species of interest. The number of columns in "..." is equal to the number of basis species (i.e., rows of `thermo()$basis`).

`...`	numeric	One or more columns of stoichiometric coefficients of basis species in the species of interest
`ispecies`	numeric	Rownumber of species in `thermo()$OBIGT`
`logact`	numeric	Logarithm of activity or fugacity of species
`state`	character	Physical state of species
`name`	character	Name of species

- `thermo()$stoich` A precalculated stoichiometric matrix for the default database. This is a matrix, not a data frame, and as such can accept duplicated row names, corresponding to chemical formulas of the species. See `retrieve`, and the first test in `inst/tinytest/test-retrieve.R` for how to update this.

`rownames`	character	Chemical formulas from `thermo()$OBIGT`
`...`	numeric	Stoichiometry, one column for each element present in any species

- `thermo()$Bdot_acirc` Values of ion size parameter (Å) for species, taken from the UT_SIZES.REF file of the HCh package (Shvarov and Bastrakov, 1999), which is based on Table 2.7 of Garrels and Christ, 1965. This is used in `nonideal` with the default '`Bdot`' method. Custom ion size parameters can be added to this vector; to override a default value for a species, either replace the numeric value for that species or prepend a named numeric value (for duplicated species, the first value is used). See `demo("yttrium")` for an example of adding and overriding species.
- `thermo()$Berman` A data frame with thermodynamic parameters for minerals in the Berman equations, assembled from files in '`extdata/Berman`' and used in `Berman`.

Note on polymorphic transitions

To enable the calculation of thermodynamic properties of polymorphic transitions, higher-temperature polymorphs of minerals are listed in OBIGT with states '`cr2`', '`cr3`', etc. The standard thermodynamic properties of high-temperature polymorphs at 25 °C and 1 bar are apparent values that are consistent with given values of enthalpy of transition (where available) at the transition temperature (T_{tr}). See the **FAQ.html** question "How can minerals with polymorphic transitions be added to the database?" for details of the retrieval of standard thermodynamic properties of polymorphs used in OBIGT.

References

Cox, J. D., Wagman, D. D. and Medvedev, V. A., eds. (1989) *CODATA Key Values for Thermodynamics*. Hemisphere Publishing Corporation, New York, 271 p. https://www.worldcat.org/oclc/18559968

Garrels, R. M. and Christ, C. L. (1965) *Solutions, Minerals, and Equilibria*, Harper & Row, New York, 450 p. https://www.worldcat.org/oclc/517586

Thoenen, T., Hummel, W., Berner, U. and Curti, E. (2014) *The PSI/Nagra Chemical Thermodynamic Database 12/07*. Paul Scherrer Institut. https://www.psi.ch/en/les/database

Wagman, D. D., Evans, W. H., Parker, V. B., Schumm, R. H., Halow, I., Bailey, S. M., Churney, K. L. and Nuttall, R. L. (1982) The NBS tables of chemical thermodynamic properties. Selected values for inorganic and C_1 and C_2 organic substances in SI units. *J. Phys. Chem. Ref. Data* **11** (supp. 2), 1–392. https://srd.nist.gov/JPCRD/jpcrdS2Vol11.pdf

See Also

Other data files, including those supporting the examples and vignettes, are documented separately at extdata.

Examples

```
## Where are the data files in CHNOSZ?
system.file("extdata", package = "CHNOSZ")
# What files make up OBIGT?
# Note: file names with _aq, _cr, _gas, or _liq
# are used in the default database
dir(system.file("extdata/OBIGT", package = "CHNOSZ"))

## Exploring thermo()$OBIGT
# What physical states are present
unique(thermo()$OBIGT$state)
# Formulas of ten random species
n <- nrow(thermo()$OBIGT)
thermo()$OBIGT$formula[runif(10)*n]

## Adding an element
old <- thermo()$element
# Element symbol, state, source (can be anything),
# mass, entropy, and number in compound
Xprops <- data.frame(element = "X", state = "cr",
  source = "user", mass = 100, s = 100, n = 1)
new <- rbind(old, Xprops)
thermo(element = new)
# Now "X" is recognized as an element in other functions
mass("X10")
# Restore default settings to remove X
reset()
```

---

util.array *Functions to Work with Multidimensional Arrays*

Description

These functions can be used to turn a list into an array and extract or replace values or take the sum along a certain dimension of an array.

Usage

```
list2array(l)
slice(arr, d = NULL, i = 1, value = NULL)
dimSums(arr, d = 1, i = NULL)
slice.affinity(affinity, d = 1, i = 1)
```

Arguments

l	a list.
arr	an array.
d	numeric, what dimension to use.
i	numeric, what slice to use.
value	values to assign to the portion of an array specified by d and i.
affinity	list, output from affinity function.

Details

list2array turns a list of arrays, each with the same dimensions, into a new array having one more dimension whose size is equal to the number of initial arrays.

slice extracts or assigns values from/to the ith slice(s) in the dth dimension of an array. Values are assigned to an array if value is not NULL. This function works by building an expression containing the extraction operator ([).

slice.affinity performs a slice operation on the 'values' element of the 'affinity' variable (which should be the output of affinity).

dimSums sums an array along the dth dimension using only the ith slices in that dimension. If i is NULL, all slices in that dimension are summed together. For matrices, dimSums(x,1) has the same result as colSums(x) and dimSums(x,2) has the same result as rowSums(x).

Examples

```
# Start with a matrix
mat <- matrix(1:12, ncol = 3)
# Pay attention to the following when
# writing examples that test for identity!
identical(1 * mat, mat)   # FALSE
# Create two matrices that are multiples of the first
a <- 1 * mat
b <- 2 * mat
# These both have two dimensions of lengths 4 and 3
dim(a)  # 4 3
# Combine them to make an array with three dimensions
x <- list2array(list(a, b))
# The third dimension has length 2
dim(x)  # 4 3 2
# The first slice of the third dimension
slice(x, 3)    # a
# The second slice of the third dimension
slice(x, 3, 2) # b
```

```
# 'slice' works just like the bracket operator
slice(x, 1)        # x[1, , ]
slice(x, 1, 2)     # x[2, , ]
slice(x, 2, 1)     # x[, 1, ]
slice(x, 2, 1:2)   # x[, 1:2, ]

# Replace part of the array
y <- slice(x, 3, 2, value = a)
# Now the second slice of the third dimension == a
slice(y, 3, 2)  # a
# and the sum across the third dimension == b
dimSums(y, 3)   # b
# Taking the sum removes that dimension
dim(y)              # 4 3 2
dim(dimSums(y, 1))  # 3 2
dim(dimSums(y, 2))  # 4 2
dim(dimSums(y, 3))  # 4 3

# Working with an 'affinity' object

basis("CHNOS+")
species("alanine")
a1 <- affinity(O2 = c(-80, -60)) # i.e. pH = 7
a2 <- affinity(O2 = c(-80, -60), pH = c(0, 14, 7))
# In the 2nd dimension (pH) get the 4th slice (pH = 7)
a3 <- slice.affinity(a2, 2, 4)
all.equal(a1$values, a3$values)  # TRUE
```

`util.data` *Functions for Checking Thermodynamic Data*

Description

Show table of references in a web browser or get individual references for species. Check self consistency of individual entries in database.

Usage

```
thermo.refs(key = NULL, keep.duplicates = FALSE)
check.EOS(eos, model, prop, return.difference = TRUE)
check.GHS(ghs, return.difference = TRUE)
check.OBIGT()
dumpdata(file)
RH2OBIGT(compound = NULL, state = "cr",
  file = system.file("extdata/adds/RH98_Table15.csv", package = "CHNOSZ"))
```

Arguments

`key`	character, numeric, or list; bibliographic reference key(s)

keep.duplicates
: logical, keep duplicated references?

eos
: dataframe, equations-of-state parameters in the format of thermo()$OBIGT

model
: character, thermodynamic model (see thermo)

prop
: character, property of interest ('Cp' or 'V')

return.difference
: logical, return the difference between database and calculated values?

ghs
: dataframe, containing G, H and S, in the format of thermo()$OBIGT

file
: character, path to a file

compound
: character, name of compound(s) in group additivity calculation

state
: character, physical state of species

Details

thermo.refs with default arguments uses browseURL to display the sources of thermodynamic data in thermo()$refs, with the URLs in that table showing as hyperlinks in the browser. Otherwise, if key is character, the citation information for those reference keys (including URLs) are returned. If key is numeric, the values refer to the species in those rows of thermo()$OBIGT, and the citation information for each listed reference (thermo()$OBIGT$ref1, thermo()$OBIGT$ref2) is returned. If key is a list, it is interpreted as the result of a call to subcrt, and the citation information for each species involved in the calculation is returned. Only unique references are returned, unless keep.duplicates is TRUE. In that case, a single reference for each species is returned, ignoring anything in thermo()$OBIGT$ref2.

check.EOS calculates heat capacity (prop = "Cp") or volume (prop = "V") from equation-of-state parameters at 25 °C and 1 bar. check.GHS calculates G (standard molal Gibbs energy of formation from the elements) from H (standard molal enthalpy of formation) and S (standard molal entropy) at 25 °C and 1 bar. The calculated values of Cp, V, or G are then compared with the given values (i.e., database values). If return.difference is TRUE (the default), the difference between the database and calculated values is returned. If return.difference is FALSE, the difference is compared with a tolerance setting (see below). If the absolute value of the difference exceeds the tolerance, the function prints a message and returns the calculated value (not the difference) of the property. If the absolute value of the difference is less than the tolerance, the function returns NA with no message.

For check.EOS, the thermodynamic parameters should be provided in eos, which is a data frame with column names in the same format as thermo$OBIGT. For check.GHS, the data frame should include G, H, S, and the chemical formula of the species. The default tolerances are 1 J/K.mol or 1 cal/K.mol for Cp (depending on the E_units for the species), 1 cm3/mol for V, and 100 cal/mol for G. These can be changed by setting thermo()optCp.tol, thermo()optV.tol, and thermo()optG.tol.

check.OBIGT is a function to check self-consistency of each entry in the thermodynamic database, using check.EOS and check.GHS. The output is a table listing only species that exceed at least one of the tolerance limits, giving the species index (rownumber in 'thermo()$OBIGT'), species name and state, and DCp, DV and DG, for the calculated differences (only those above the tolerances are given). Values of DCp and DG are given in the units present in the data files. This function is used to generate the file found at extdata/thermo/OBIGT_check.csv.

dumpdata returns all of the available data, from both the default and optional data files, or writes it to a file if file is not NULL. The format is the same as thermo$OBIGT, except for a single prepended column named 'source', giving the source of the data ('OBIGT' refers to the default database, and 'DEW', 'SLOP98', and 'SUPCRT92' are the optional data files).

RH2OBIGT implements a group additivity algorithm for standard molal thermodynamic properties and equations of state parameters of crystalline and liquid organic molecules from Richard and Helgeson, 1998. The names of the compounds and their physical state are searched for in the indicated file, that also contains chemical formulas and group stoichiometries; the names of the groups are stored in the column names of this file, and must be present in thermo$OBIGT. The default file (extdata/thermo/RH98_Table15.csv) includes data taken from Table 15 of Richard and Helgeson, 1998 for high molecular weight compounds in 'cr'ystalline and 'liq'uid states. An error is produced if any of the compound-state combinations is not found in the file, if any of the group names for a given compound-state combination is not found in thermo()$OBIGT, or if the chemical formula calculated from group additivity (with the aid of i2A and as.chemical.formula) is not identical to that listed in the file.

Value

The values returned (invisible-y) by mod.OBIGT are the rownumbers of the affected species.

References

Richard, L. and Helgeson, H. C. (1998) Calculation of the thermodynamic properties at elevated temperatures and pressures of saturated and aromatic high molecular weight solid and liquid hydrocarbons in kerogen, bitumen, petroleum, and other organic matter of biogeochemical interest. *Geochim. Cosmochim. Acta* **62**, 3591–3636. doi:10.1016/S00167037(97)003451

See Also

thermo, add.OBIGT, mod.buffer

Examples

```
# Citation information for Helgeson et al., 1998
thermo.refs("HOK+98")
# Two references for alanine
thermo.refs(info("alanine"))
# Three references for species in the reaction
s <- subcrt(c("O2","O2"),c("gas","aq"),c(-1,1))
thermo.refs(s)
## Not run:
## Marked dontrun because it opens a browser
# Show the contents of thermo()$refs
thermo.refs()

## End(Not run)

## Calculate thermodynamic properties of organic compounds
## using group additivity, after Richard and Helgeson, 1998
RH2OBIGT()
```

util.expression *Functions to Express Chemical Formulas and Properties*

Description

Generate expressions suitable for axis labels and plot legends describing chemical species, properties and reactions.

Usage

```
expr.species(species, state = "aq", value=NULL, log=FALSE, molality=FALSE,
  use.state=FALSE, use.makeup=FALSE)
expr.property(property, molality = FALSE)
expr.units(property, prefix = "", per = "mol")
axis.label(label, units = NULL, basis = thermo()$basis, prefix = "",
  molality = FALSE)
describe.basis(ibasis = 1:nrow(basis), basis = thermo()$basis,
  digits = 1, oneline = FALSE, molality = FALSE, use.pH = TRUE)
describe.property(property, value, digits = 0, oneline = FALSE,
  ret.val = FALSE)
describe.reaction(reaction, iname = numeric(), states = NULL)
syslab(system = c("K2O", "Al2O3", "SiO2", "H2O"), dash="-")
ratlab(top = "K+", bottom = "H+", molality = FALSE)
```

Arguments

species	character, formula of a chemical species
state	character, designation of physical state
value	numeric, logarithm of activity or fugacity of species, or value of other property
log	logical, write logarithm of activity/fugacity/molality?
molality	logical, use molality (m) instead of activity (a) for aqueous species?
use.state	logical, include state in expression?
use.makeup	logical, use makeup to count the elements?
use.pH	logical, use pH instead of log activity of H+?
property	character, description of chemical property
prefix	character, prefix for units
per	character, denominator in units
label	character, description of species, condition or property
units	character, description of units
ibasis	numeric, which basis species to include
basis	data frame, definition of basis species
digits	numeric, number of digits to show after decimal point

oneline	logical, make descriptions occupy a single line?
ret.val	logical, return only the value with the units?
reaction	data frame, definition of reaction
iname	numeric, show names instead of formulas for these species
states	character, if 'all', show states for all species; numeric, which species to show states for
system	character, thermodynamic components
dash	character to use for dash between components
top	character, the ion in the numerator of the ratio
bottom	character, the ion in the denominator of the ratio

Details

The expr.* functions create expressions using the plotmath syntax to describe the names and states and logarithms of activity or fugacity of chemical species, conditions including temperature and pressure and chemical properties such as Gibbs energy and volume.

expr.species constructs a formatted expression using the formula or name of a single chemical species. With no other arguments, the formula is just formatted with the appropriate subscripts and superscripts. Providing the physical state adds a variable to the expression (*a* for aqueous species and pure phases, except *f* for gases). Set molality to TRUE to write *m* instead of *a* for aqueous species. The state itself is written in the expression if use.state is TRUE. If log is TRUE, the expression includes a 'log' prefix. Finally, provide a value in value to write an equation (something like logfO2 = -70), or set it to NA to only write the variable itself (e.g. logfO2). Set use.makeup to TRUE to use makeup to parse the chemical formula. This was an older default action that had the undesirable effect of reordering and grouping all the elements, and has been replaced with a different splitting algorithm so that coefficients and charges are sub/superscripted without affecting the intervening text.

expr.property accepts a description in property that indicates the chemical property of interest. Uppercase letters are italicized, and lowercase letters are italicized and subscripted. Other specific characters are parsed as follows (case-sensitive):

'D'	Delta
'A'	bold A (chemical affinity)
'p'	subscript italic P (for isobaric heat capacity)
'0'	degree sign (for a standard-state property)
'l'	subscript lambda
'''	prime symbol

A '0' gets interpreted as a degree sign only if it does not immediately follow a number (so that e.g. '2.303' can be included in an expression).

Every other character that is one of the letters or LETTERS in the description of the property is italicized in the expression; other characters such as numerals or mathematical operators are shown without any special formatting. Special cases for the property argument ('logK', 'Eh', 'pH', 'pe', 'IS' and 'ZC') are interpreted as simple expressions, and are not parsed according to the

above rules.

`expr.units` returns an expression for the units, based on one or more characters appearing in the `property`:

'A', 'G', 'H'	energy
'Cp', 'S'	energy per Kelvin
'V'	volume
'E'	volume per Kelvin
'P'	pressure
'T'	temperature
'Eh'	electrical potential
'IS'	ionic strength

If none of those characters appears in the property, the expression is an empty character (no units). If a `prefix` is given, it is added to the expression. The denominator of the units (default '`mol`') is taken from the `per` argument; it is applied to all units except for '`P`', '`T`', '`Eh`', and '`IS`'.

`axis.label` accepts a generic description of a `label`. If this matches the chemical formula of one of the basis species in the `basis` argument, the expression for the label is generated using `expr.species` with `log` set to the physical state of the basis species. Otherwise, the expression is built by combining the output of `expr.property` with `expr.units` (or the value in `units`, if it is supplied), placing a comma between the two. This function is used extensively in `diagram` and also appears in many of the examples. Note that `diagram` sets `molality` to TRUE if `IS` was supplied as an argument to `affinity`.

`describe.basis` makes an expression summarizing the basis species definition (logarithms of activity or fugacity of the basis species) provided in `basis`; only the basis species identified by `ibasis` are included.

`describe.property` makes an expression summarizing the properties supplied in `property`, along with their `values`. The expressions returned by both functions consist of a property, an equals sign, and a value (with units where appropriate); the expressions have a length equal to the number of property/value pairs. If `oneline` is TRUE, the property/value pairs are combined into a single line, separated by commas. The number of digits shown after the decimal point in the values is controlled by `digits`. If `ret.val` is TRUE, only the values and their units are returned; this is useful for labeling plots with values of temperature.

`describe.reaction` makes an expression summarizing a chemical reaction. The `reaction` data frame can be generated using `subcrt`. Based on the sign of their reaction coefficients, species are placed on the reactant (left) or product (right) side of the reaction, where the species with their coefficients are separated by plus signs; the two sides of the reaction are separated by a reaction double arrow (Unicode U+21CC). Coefficients equal to 1 are not shown. Chemical formulas of species include the physical state if `states` is '`all`', or a numeric value indicating which species to label with the state. Names of species (as provided in `reaction`) are shown instead of chemical formulas for the species identified by `iname`.

`syslab` formats the given thermodynamic components (using `expr.species`) and adds intervening en dashes.

`ratlab` produces a expression for the activity ratio between the ions in the `top` and `bottom` arguments. The default is a ratio with H^+, i.e. (activity of the ion) / [(activity of H+) ^ (charge of the ion)].

See Also

util.legend for other functions to make legends. demo("saturation") for examples of syslab and ratlab.

Examples

```
## Show descriptions of species and properties on a plot
plot(0, 0, xlim = c(1,5), ylim = c(1,5), xlab = "function", ylab = "example")
text0 <- function(...) text(..., adj = 0)
# Species
text0(1, 1, expr.species("CO2"))
text0(1, 2, expr.species("CO2", use.state = TRUE))
text0(1, 3, expr.species("CO2", log = TRUE, use.state = TRUE))
text0(1, 4, expr.species("CO2", log = TRUE))
text0(1, 5, expr.species("CO2", log = TRUE, value = -3))
# Properties
text0(2, 1, expr.property("A"))
text0(2, 2, expr.property("DV"))
text0(2, 3, expr.property("DG0f"))
text0(2, 4, expr.property("DCp0,r"))
text0(2, 5, expr.property("T"))
# Units
text0(3, 1, expr.units("A", prefix = "k"))
text0(3, 2, expr.units("DV"))
text0(3, 3, expr.units("DG0f", prefix = "k"))
text0(3, 4, expr.units("DCp0,r"))
text0(3, 5, expr.units("T"))
# axis.label
text0(4, 1, axis.label("DG0f"))
text0(4, 2, axis.label("T"))
text0(4, 3, axis.label("pH"))
text0(4, 4, axis.label("Eh"))
text0(4, 5, axis.label("IS"))
# describe.basis
basis("CHNOS+")
dbasis <- describe.basis(oneline = TRUE, digits = 0)
property <- c("P", "T", "Eh", "pH", "IS")
value <- c(1, 42.42, -1, 7, 0.1)
dprop <- describe.property(property, value, oneline = TRUE)
text(3, 1.5, dbasis)
text(3, 2.5, dprop)
dbasis <- describe.basis(c(1, 5))
dprop <- describe.property(property[1:2], value[1:2])
legend(2.4, 3.9, legend=c(dbasis, dprop), bty = "n")
# describe.reaction
# Reaction is automatically balanced because basis species are defined
reaction <- subcrt("glucose", -1)$reaction
text(3, 4.25, describe.reaction(reaction))
text(3, 4.5, describe.reaction(reaction, states = "all"))
text(3, 4.75, describe.reaction(reaction, iname = 1:4))
title(main = "Plot labels for chemical species and thermodynamic properties")
```

util.fasta *Functions for Reading FASTA Files and Downloading from UniProt*

Description

Search the header lines of a FASTA file, read protein sequences from a file, count numbers of amino acids in each sequence, and download sequences from UniProt.

Usage

```
read.fasta(file, iseq = NULL, ret = "count", lines = NULL,
  ihead = NULL, start=NULL, stop=NULL, type="protein", id = NULL)
count.aa(seq, start=NULL, stop=NULL, type="protein")
```

Arguments

file	character, path to FASTA file
iseq	numeric, which sequences to read from the file
ret	character, specification for type of return (count, sequence, or FASTA format)
lines	list of character, supply the lines here instead of reading them from file
ihead	numeric, which lines are headers
start	numeric, position in sequence to start counting
stop	numeric, position in sequence to stop counting
type	character, sequence type (protein or DNA)
id	character, value to be used for protein in output table
seq	character, amino acid sequence of a protein

Details

read.fasta is used to retrieve entries from a FASTA file. Use iseq to select the sequences to read (the default is all sequences). The function returns various formats depending on the value of ret. The default 'count' returns a data frame of amino acid counts (the data frame can be given to add.protein in order to add the proteins to thermo$protein), 'seq' returns a list of sequences, and 'fas' returns a list of lines extracted from the FASTA file, including the headers (this can be used e.g. to generate a new FASTA file with only the selected sequences). If the line numbers of the header lines were previously determined, they can be supplied in ihead. Optionally, the lines of a previously read file may be supplied in lines (in this case no file is needed so file should be set to ""). When ret is 'count', the names of the proteins in the resulting data frame are parsed from the header lines of the file, unless id is provided. If id is not given, and a UniProt FASTA header is detected (regular expression "\|.....\|.*_"), information there (accession, name, organism) is split into the protein, abbrv, and organism columns of the resulting data frame.

count.aa counts the occurrences of each amino acid or nucleic-acid base in a sequence (seq). For amino acids, the columns in the returned data frame are in the same order as thermo()$protein.

The matching of letters is case-insensitive. A warning is generated if any character in `seq`, excluding spaces, is not one of the single-letter amino acid or nucleobase abbreviations. `start` and/or `stop` can be provided to count a fragment of the sequence (extracted using `substr`). If only one of `start` or `stop` is present, the other defaults to 1 (`start`) or the length of the sequence (`stop`).

Value

`read.fasta` returns a list of sequences or lines (for `ret` equal to '`seq`' or '`fas`', respectively), or a data frame with amino acid compositions of proteins (for `ret` equal to '`count`') with columns corresponding to those in `thermo$protein`.

See Also

`seq2aa`, like `count.aa`, counts amino acids in a user-input sequence, but returns a data frame in the format of `thermo()$protein`.

Examples

```
## Reading a protein FASTA file
# The path to the file
file <- system.file("extdata/protein/EF-Tu.aln", package = "CHNOSZ")
# Read the sequences, and print the first one
read.fasta(file, ret = "seq")[[1]]
# Count the amino acids in the sequences
aa <- read.fasta(file)
# Compute lengths (number of amino acids)
protein.length(aa)

## Not run:
## Count amino acids in a sequence
count.aa("GGSGG")
# Warnings are issued for unrecognized characters
atest <- count.aa("WhatAmIMadeOf?")
# There are 3 "A" (alanine)
atest[, "A"]

## End(Not run)
```

---

util.formula *Functions to Work with Chemical Formulas*

Description

Calculate the standard molal entropy of elements in a compound; calculate the standard molal Gibbs energy or enthalpy of formation, or standard molal entropy, from the other two; list coefficients of selected elements in a chemical formula; calculate the average oxidation state of carbon. Create a stoichiometric matrix for selected species.

Usage

```
as.chemical.formula(makeup, drop.zero = TRUE)
mass(formula)
entropy(formula)
GHS(formula, G = NA, H = NA, S = NA, T = 298.15, E_units = "J")
ZC(formula)
i2A(formula)
```

Arguments

makeup	numeric, object returned by makeup
drop.zero	logical, drop elements with a coefficient of zero?
formula	character, chemical formulas, or numeric, rownumbers in thermo()$OBIGT
G	numeric, standard molal Gibbs energy of formation from the elements
H	numeric, standard molal enthalpy of formation from the elements
S	numeric, standard molal molal entropy
T	numeric, temperature in Kelvin
E_units	character, energy units (J or cal)

Details

i2A returns a stoichiometric matrix representing the elemental composition of the formulas. Each column corresponds to an element that is present in at least one of the formulas; some element counts will be zero if not all formula have the same elements. If a matrix is passed to i2A it is returned unchanged.

as.chemical.formula makes a character string representing a chemical formula from a vector of coefficients with names corresponding to the elements (e.g., the output of makeup) or from a stoichiometric matrix (output of i2A). Each elemental symbol is written followed by its coefficient; negative coefficients are signed. Any coefficients equal to 1 are not explicitly written, and any charge (indicated by makeup as 'Z') is shown as a signed number at the end of the formula. If the formula is uncharged, and the last element has a negative coefficient, +0 is shown at the end of the formula to indicate a charge of zero.

The remaining functions documented here accept vectors of chemical formulas, species indices, or a mixture of both, or stoichiometric matrices with elements on the columns.

mass and entropy return the sums of masses or entropies of elements in each of the formulas. The masses are calculated using the masses of the elements in their natural isotopic distribution, and the entropies, in J K^{-1} mol^{-1}, are calculated using the entropies of the compounds of the pure elements in their stable states at 25 °C and 1 bar. The properties of the elements used by this function are taken from thermo$element.

GHS computes one of the standard molal Gibbs energy or enthalpy of formation from the elements, or standard molal entropy, from values of the other two. The formula, G, H and S arguments must all have the same length. The entropies of the elements (Se) in each formula are calculated using entropy, which gives values in Joules. If E_units is 'cal', the values are converted to calories. The equation in effect can be written as $\Delta G^\circ = \Delta H^\circ - T\Delta S^\circ$, where $\Delta S^\circ = S - S_e$ and T is the temperature given in T (defaults to 298.15 K) (note that G and H in the arguments correspond

respectively to ΔG° and ΔH° in the equation). For each formula, if one of `G`, `H`, or `S` is NA, its value is calculated from the other two. Otherwise, the values are returned unchanged.

`ZC` returns the average oxidation state of carbon (Z_C) calculated from ratios of the elements in the chemical `formulas`. The equation used is $Z_C = \frac{Z - n_H + 2(n_O + n_S) + 3n_N}{n_C}$, where the n refer to the number of the indicated element in the formula and Z is the charge (Dick and Shock, 2011). The result is `NaN` for any formula that does not contain carbon. Elements other than those shown in the equation are not included in the calculation, and produce a warning.

Value

`mass`, `entropy`, and `ZC` return numeric values. `as.chemical.formula` returns a character object. `GHS` returns a matrix with column names 'G', 'H' and 'S', and `i2A` returns a matrix with column names corresponding to the elements in the formulas.

References

Dick, J. M. and Shock, E. L. (2011) Calculation of the relative chemical stabilities of proteins as a function of temperature and redox chemistry in a hot spring. *PLOS One* **6**, e22782. doi:10.1371/journal.pone.0022782

See Also

`makeup`, used by `mass` and `entropy`, and `ZC` and `i2A` for counting the elements in a formula (the latter two make use of the `count.zero` argument).

Examples

```
## Mass and entropy from chemical formulas
mass("H2O")
entropy("H2O")
mass("-1")  # electron
entropy("-1")

## Different ways to get the formula of alanine
iA <- info("alanine")
info(iA)$formula
as.chemical.formula(makeup(iA))

## Converting among Gibbs energy, enthalpy, entropy
# Calculate the value of G from H and S
GHS("H2O", H = water("H")$H, S = water("S")$S)
# The result is not exactly equal to the value from water("G"),
# probably because of different entropies of the elements

## Average oxidation states of carbon
ZC(c("CO2", "CH4", "CHNOSZ"))  # 4, -4, 7
si <- info("LYSC_CHICK")
# Can use species index or formula
ZC(si)
ZC(info(si)$formula)
```

```
## Calculate the chemical formulas, then
## ZC of all of the proteins in CHNOSZ's database
pf <- protein.formula(thermo()$protein)
range(mass(pf))
# Use na.rm = TRUE because we have a "protein" with a formula of H2O
range(ZC(pf), na.rm = TRUE)
```

`util.legend` *Functions to Make Legend Text*

Description

Generate expressions describing system conditions that can be added to legends of plots.

Usage

```
lNaCl(x, digits = 2)
lS(x, digits = 3)
lT(x, digits = 0)
lP(x, digits = 0)
lTP(x, y, digits = 0)
lex(...)
```

Arguments

`x`	numeric, value of the property
`digits`	numeric, digits for rounding
`y`	numeric, value of pressure
`...`	language, objects to combine in an expression

Details

These functions are used to make expressions for common chemical system variables that can be used in plot legends. `lNaCl` describe the molality of NaCl, and `lS` the total molality of sulfur. `lT` and `lP` describe the temperature and pressure. `lTP` describe the temperature and pressure together, separated by a comma.

The above functions return language objects, which can be combined with `lex` to make an expression that when used in `legend` appears on multiple lines.

See Also

`util.expression`

Examples

```
plot.new()
l <- lex(lTP(100, "Psat"), lNaCl(1), lS(1e-3))
legend("center", l)
```

util.list *Functions to Work with Lists*

Description

Identify list elements that have the maximum (or minimum) values.

Usage

```
which.pmax(x, maximum = TRUE)
```

Arguments

x	list of numeric vectors
maximum	logical, find maximum values or minimum values?

Details

which.pmax takes a list of equal-length numeric vectors or equal-dimension arrays in x and returns the index of the list element that has the maximum value at each point. Change maximum to FALSE to find the minimum values instead.

util.misc *Functions for Miscellaneous Tasks*

Description

Calculate dP/dT and temperature of polymorphic transitions; scale logarithms of activity to a desired total activity.

Usage

```
dPdTtr(ispecies, ispecies2 = NULL)
Ttr(ispecies, ispecies2 = NULL, P = 1, dPdT = NULL)
GHS_Tr(ispecies, Htr)
unitize(logact = NULL, length = NULL, logact.tot = 0)
```

Arguments

ispecies	numeric, species index of a mineral phase
ispecies2	numeric, species index of next mineral phase (the default is ispecies + 1)
P	numeric, pressure (bar)
dPdT	numeric, values of (dP/dT) of polymorphic transitions (Ttr)
Htr	numeric, enthalpy(ies) of transition (cal/mol)
logact	numeric, logarithms of activity
length	numeric, numbers of residues
logact.tot	numeric, logarithm of total activity

Details

dPdTtr returns values of $(dP/dT)_{Ttr}$, where Ttr represents the transition temperature, of the polymorphic transition at the high-T stability limit of the ispecies in thermo()$OBIGT (other than checking that the names match, the function does not check that the species in fact represent different phases of the same mineral). dPdTtr takes account of the Clapeyron equation, $(dP/dT)_{Ttr}$=$\Delta S/\Delta V$, where ΔS and ΔV represent the changes in entropy and volume of polymorphic transition, and are calculated using subcrt at Ttr from the standard molal entropies and volumes of the two phases involved. Using values of dPdT calculated using dPdTtr or supplied in the arguments, Ttr returns as a function of P values of the upper transition temperature of the mineral phase represented by ispecies.

GHS_Tr can be used to calculate values of G, H, and S at Tr for the cr2, cr3, and cr4 phases in the database. It combines the given Htr (enthalpies of transition) with the database values of GHS @ Tr only for the phase that is stable at 298.15 K (cr) and the transition temperatures and Cp coefficients for higher-temperature phases, to calculate the GHS @ Tr (i.e. low-temperature metastable conditions) of the phases that are stable at higher temperatures.

unitize scales the logarithms of activities given in logact so that the logarithm of total activity of residues is equal to zero (i.e. total activity of residues is one), or to some other value set in logact.tot. length indicates the number of residues in each species. If logact is NULL, the function takes the logarithms of activities from the current species definition. If any of those species are proteins, the function gets their lengths using protein.length.

Examples

```
# We need the Helgeson et al., 1978 minerals for this example
add.OBIGT("SUPCRT92")
# That replaces the existing enstatite with the first phase;
# the other phases are appended to the end of thermo()$OBIGT
i1 <- info("enstatite")
i2 <- info("enstatite", "cr2")
i3 <- info("enstatite", "cr3")
# (dP/dT) of transitions
dPdTtr(i1, i2)  # first transition
dPdTtr(i2, i3)  # second transition
# Temperature of transitions (Ttr) as a function of P
Ttr(i1, i2, P = c(1,10,100,1000))
Ttr(i2, i3, P = c(1,10,100,1000))
# Restore default database
OBIGT()

# Calculate the GHS at Tr for the high-temperature phases of iron
# using transition enthalpies from the SUPCRT92 database (sprons92.dat)
Htr <- c(326.0, 215.0, 165.0)
iiron <- info("iron")
GHS_Tr(iiron, Htr)
# The results calculated above are stored in the database ...
info(1:3 + iiron)[, c("G", "H", "S")]
# ... meaning that we can recalculate the transition enthalpies using subcrt()
sapply(info(0:2 + iiron)$T, function(T) {
  # A very small T increment around the transition temperature
  T <- convert(c(T-0.01, T), "C")
```

```
  # Use suppressMessages to make the output less crowded
  sres <- suppressMessages(subcrt("iron", T = T, P = 1))
  diff(sres$out$iron$H)
})

## Scale logarithms of activity
# Suppose we have two proteins whose lengths are 100 and
# 200; what are the logarithms of activity of the proteins
# that are equal to each other and that give a total
# activity of residues equal to unity?
logact <- c(-3, -3)  # could be any two equal numbers
length <- c(100, 200)
logact.tot <- 0
loga <- unitize(logact, length, logact.tot)
# The proteins have equal activity
loga[1] == loga[2]
# The sum of activity of the residues is unity
all.equal(sum(10^loga * length), 1)
## What if the activity of protein 2 is ten times that of protein 1?
logact <- c(-3, -2)
loga <- unitize(logact, length, logact.tot)
# The proteins have unequal activity,
# but the activities of residues still add up to one
all.equal(loga[2] - loga[1], 1)
all.equal(sum(10^loga * length), 1)
```

`util.plot` *Functions to Create and Modify Plots*

Description

Initialize a new plot window using preset parameters, add an axis or title to a plot, generate labels for axes and subplots, add stability lines for water, get colors for a set of numeric values.

Usage

```
thermo.plot.new(xlim, ylim, xlab, ylab, cex = par("cex"),
  mar = NULL, lwd = par("lwd"), side = c(1,2,3,4),
  mgp = c(1.7, 0.3, 0), cex.axis = par("cex"), col = par("col"),
  yline = NULL, axs = "i", plot.box = TRUE, las = 1,
  xline = NULL, grid = "", col.grid = "gray", ...)
thermo.axis(lab = NULL, side = 1:4, line = 1.5, cex = par("cex"),
  lwd = par("lwd"), col = par("col"), grid = "", col.grid = "gray",
  plot.line = FALSE)
label.plot(x, xfrac = 0.07, yfrac = 0.93, paren = FALSE,
  italic = FALSE, ...)
usrfig()
label.figure(x, xfrac = 0.05, yfrac = 0.95, paren = FALSE,
  italic = FALSE, ...)
```

```
water.lines(eout, which = c("oxidation","reduction"),
  lty = 2, lwd = 1, col = par("fg"), plot.it = TRUE)
mtitle(main, line = 0, spacing = 1, ...)
ZC.col(z)
add.alpha(col, alpha)
```

Arguments

xlim	numeric, limits of the x-axis
ylim	numeric, limits of the y-axis
xlab	character, x-axis label
ylab	character, y-axis label
cex	numeric, character expansion factor for labels
mar	numeric, width (number of lines) of margins on each side of plot
lwd	numeric, line width
side	numeric, which sides of plot to draw axes
mgp	numeric, sizes of margins of plot
cex.axis	numeric, character expansion factor for names of axes
col	character, color
yline	numeric, margin line on which to plot y-axis name
axs	character, setting for axis limit calculation
plot.box	logical, draw a box around the plot?
las	numeric, style for axis labels
xline	numeric, margin line on which to plot x-axis name
grid	character, type of grid ('major', 'minor', or 'both')
col.grid	character, color of the grid lines
plot.line	logical, draw axis lines?
...	further arguments passed to par or mtext
lab	character, axis label
line	numeric, margin line on which to place axis label or plot title
x	character, label to place on plot
xfrac	numeric, fractional location on x-axis for placement of label
yfrac	numeric, fractional location on y-axis for placement of label
paren	logical, add parentheses around label text?
italic	logical, italicize label text?
eout	data frame, output of affinity, equilibrate, or diagram
which	character, which of oxidation/reduction lines to plot
lty	numeric, line type
plot.it	logical, plot the lines?
main	character, text for plot title
spacing	numeric, spacing between multiple lines
z	numeric, set of values
alpha	character, hexadecimal value of color transparency (alpha)

Details

thermo.plot.new sets parameters for a new plot, creates a new plot using plot.new, and adds the axes tick marks to the plot. Plot parameters (see par) including cex, mar, lwd, mgp and axs can be given, as well as a numeric vector in side identifying which sides of the plot receive tick marks. yline, if present, denotes the margin line (default par('mgp')[1]) where the y-axis name is plotted. thermo.axis is the function that actually adds the axes, including inward-pointing major and minor tick marks (often used for thermodynamic property diagrams).

Use grid to add a grid to the plot, corresponding to either the major ticks (solid lines), minor ticks (dashed lines), or both. The grid can be made by adding grid argument to diagram, or by calling thermo.axis after diagram (see example).

water.lines plots lines representing the oxidation and reduction stability limits of water on Eh/pe/log f_{O_2}/log f_{H_2} vs pH/T/P diagrams. The x- and y-variables and their ranges are taken from eout. Values of T, P, pH, and $\log a_{H_2O}$, not corresponding to either axis, are also taken from eout. which controls which lines are drawn ('oxidation', 'reduction', or both (the default)). The value of swapped in the output reflects whether pH, T, or P is on the x-axis (TRUE) or y-axis (FALSE). NA is returned for any diagram for variables that can not be processed (including diagrams with more than 2 variables).

label.plot and label.figure add identifying text within the plot region and figure region. The value given for x is made into a label, optionally italicized and with parentheses (like (a)). The location of the label is controlled by xfrac and yfrac (the fractional coordinates of either the plot or figure region), and ... can include other parameters such as cex and adj that are passed to text.

usrfig returns the limits of the figure region in "user" coordinates (i.e. the limits of the plot region, from par("usr")). It is a supporting function for label.figure but is also useful for other circumstances where information must be added at a particular location in a figure.

mtitle can be used to add a multi-line title to a plot. It loops over each element of main and places it on a separate margin line using mtext. The spacing of the last (bottom) line from the edge of the plot is specified by line. This function exists to facilitate using expressions in multiline titles.

ZC.col uses **colorspace**) to generate colors from a diverging palette (red - light grey - blue) corresponding to the values in z. Red is associated with lower values of z. This function is intended to generate colors for distinguishing average oxidation state of carbon ZC, but any numeric values can be supplied.

add.alpha adds transparency to a color by appending the value of alpha to the hexadecimal representation of the color given in col.

See Also

diagram uses thermo.plot.new to set up a new plot, unless the argument tplot is set to FALSE in diagram.

Examples

```
basis(c("H2S", "H2O", "H+", "e-"))
species(c("HS-", "H2S", "HSO4-", "SO4-2"))
a <- affinity(pH = c(0, 12), Eh = c(-1, 1), T = 200)
```

```
opar <- par(mfrow = c(2, 2))
diagram(a, grid = "both")
title(main = 'diagram(a, grid = "both")')
diagram(a, grid = "major")
title(main = 'diagram(a, grid = "major")')
diagram(a, grid = "minor")
title(main = 'diagram(a, grid = "minor")')
diagram(a, fill = "terrain")
thermo.axis(grid = "major", col.grid = "slategray")
title(main = 'thermo.axis(grid = "major")')
par(thermo()$opar)
par(opar)
```

`util.protein` *Functions for Proteins (Other Calculations)*

Description

Return chemical formulas of groups in proteins, and calculate heat capacity using an additivity model from the literature.

Usage

```
MP90.cp(protein, T)
group.formulas()
```

Arguments

`protein`	proteins specified in any format usable by `pinfo`
`T`	numeric, temperature in °C

Details

`group.formulas` returns the chemical formulas of each of the 20 common amino acid residues in proteins, as well as the terminal -H and -H (treated as the [H2O] group).

`MP90.cp` takes `protein` (name of protein) and `T` (one or more temperatures in °C and returns the additive heat capacity (J mol $^{-1}$) of the unfolded protein using values of heat capacities of the residues taken from Makhatadze and Privalov, 1990. Those authors provided values of heat capacity at six points between 5 and 125 °C; this function interpolates (using `splinefun`) values at other temperatures.

References

Makhatadze, G. I. and Privalov, P. L. (1990) Heat capacity of proteins. 1. Partial molar heat capacity of individual amino acid residues in aqueous solution: Hydration effect *J. Mol. Biol.* **213**, 375–384. doi:10.1016/S00222836(05)801974

See Also

ionize.aa for an example that compares MP90.cp with heat capacities calculated in CHNOSZ at different temperatures and pHs.

---

util.seq *Functions to Work with Sequence Data*

Description

Return names or one- or three-letter abbreviations of amino acids.

Usage

```
aminoacids(nchar = 1, which = NULL)
```

Arguments

nchar	numeric, 1 to return one-letter, 3 to return three-letter abbreviations for amino acids
which	character, which amino acids to name

Details

aminoacids returns the one-letter abbreviations (nchar='1') or the three-letter abbreviations (nchar='3') or the names of the neutral amino acids (nchar='""') or the names of the amino acids with ionized side chains (nchar='"Z"'). The output includes 20 amino acids in alphabetic order by 1-letter abbreviation (the order used in thermo()$protein), unless which is provided, indicating the desired amino acids (either as 1- or 3-letter abbreviations or names of the neutral amino acids).

See Also

count.aa for counting amino acids or nucleic-acid bases in a sequence; protein.formula for calculating the chemical formulas of proteins.

Examples

```
## Count nucleobases in a sequence
bases <- count.aa("ACCGGGTTT", type = "DNA")
```

util.units *Functions to Convert Units*

Description

These functions convert values between units and set the user's preferred units.

Usage

```
P.units(units = NULL)
T.units(units = NULL)
E.units(units = NULL)
convert(value, units, T = 298.15, P = 1, pH = 7, logaH2O = 0)
```

Arguments

units	character, name of units to set or convert to/from
value	numeric, value(s) to be converted
T	numeric, temperature (Kelvin), used in 'G'-'logK', 'pe'-'Eh' and 'logfO2'-'E0' conversions
P	numeric, pressure (bar), used in 'logfO2'-'E0' conversions
pH	numeric, pH, used in 'logfO2'-'E0' conversions
logaH2O	numeric, logarithm of activity of water, used in 'logfO2'-'E0' conversions

Details

The units settings are used by subcrt, affinity, and diagram to accept input in or convert output to the units desired by the user. The settings, which can be queried or changed with T.units, E.units and P.units, refer to the units of temperature (C or K), energy (J or cal), and pressure (bar or MPa). (The first value in each of those pairs refers to the default units).

The actual units conversions are handled by convert, through which values are transformed into destination units (names not case sensitive). The possible conversions and settings for the units argument are shown in the following table. Note that 'Eh' and 'E0' both stand for the value of Eh (oxidation-reduction potential in volts); they have different names so that one can choose to convert between Eh and either 'pe' or 'logfO2'.

property	units	setting of units argument
temperature	°C, K	C, K
pressure	bar, MPa	bar, MPa
energy	cal, J	cal, J
energy	J, cm^3 bar	joules, cm3bar
energy	J, [none]	G, logK
oxidation potential	volt, [none]	Eh, pe
oxidation potential	volt, [none]	E0, logfO2

Another use of the function is to convert the results from `solubility` into parts per billion, million, or thousand. These destination units are specified by 'ppb', 'ppm', or 'ppt'. Additionally, the logarithms can be chosen with 'logppb', 'logppm', and 'logppt'. See `demo("contour")` and `demo("sphalerite")` for examples.

Examples

```
## Direct usage of convert
# Temperature (Kelvin) to degrees C
convert(273.15, "C")
# Temperature (degrees C) to Kelvin
convert(100, "K")
# Gibbs energy (J mol-1) to/from logK
convert(1000, "logK")
convert(1000, "logK", T = 373.15)
convert(1, "G")
# Eh (volt) to pe
convert(-1, "pe")
convert(-1, "pe", T = 373.15)
# logfO2 to E0 (volt)
convert(-80, "E0")
convert(-80, "E0", pH = 5)
convert(-80, "E0", pH = 5, logaH2O = -5)
# Convert from calories to Joules
convert(1, "J")    # 1 cal = 4.184 J
# Convert from Joules to calories
convert(1, "cal")  # 1 J = 0.239 cal
# Convert cm3bar to Joules
convert(10, "joules") # 10 cm3.bar = 1 J

## Setting the units
# Make K the units for temperature arguments to subcrt() and affinity()
T.units("K")
# Return to default - degrees C
T.units("C")
```

`util.water` *Functions for Properties of Water and Steam*

Description

Utility functions for properties of water and steam.

Usage

```
WP02.auxiliary(property, T = 298.15)
rho.IAPWS95(T = 298.15, P = 1, state="", trace=0)
water.AW90(T = 298.15, rho = 1000, P = 0.1)
```

Arguments

property	character, property to calculate
T	numeric, temperature (K)
P	numeric, pressure (units of bar, except MPa for water.AW90)
state	character, state or phase of H_2O
trace	integer number
rho	numeric, density (kg m^{-3})

Details

Auxiliary equations to the IAPWS-95 formulation (Wagner and Pruß, 2002) are provided in WP02.auxiliary. The property for this function can be one of 'P.sigma' (saturation vapor pressure in MPa), 'dP.sigma.dT' (derivative of saturation vapor pressure with respect to temperature), or 'rho.liquid' or 'rho.vapor' (density of liquid or vapor in kg m^{-3}).

rho.IAPWS95 implements a root-finding technique (using uniroot) to determine the values of density for the stable phase of H_2O at the given temperature and pressure. The state option is used internally in order to determine the stable phase at conditions close to saturation (0.9999*P_{SAT} <= P <= 1.00005*P_{SAT}, where P_{SAT} is the saturation pressure calculated by WP02.auxiliary). Alternatively, the user can specify a state of 'liquid' or 'vapor' to force the calculation of density for the corresponding phase, even if it is metastable (e.g. superheated water, supercooled steam; this option has no effect in the supercritical region). The state is set in calls by water.IAPWS95 to the value in thermo()optIAPWS.sat (default 'liquid') so that higher-level functions (water, subcrt) take properties for that state along the saturation curve. Diagnostic messages are printed if trace is positive (it is also included in the call to uniroot).

water.AW90 provides values of the static dielectric constant (epsilon) calculated using equations given by Archer and Wang, 1990.

References

Archer, D. G. and Wang, P. M. (1990) The dielectric constant of water and Debye-Hückel limiting law slopes. *J. Phys. Chem. Ref. Data* **19**, 371–411. https://srd.nist.gov/JPCRD/jpcrd383.pdf

Wagner, W. and Pruß, A. (2002) The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *J. Phys. Chem. Ref. Data* **31**, 387–535. doi:10.1063/1.1461829

Examples

```
# Calculate density of stable phase at 500 K, 500 bar
rho <- rho.IAPWS95(T = 500, P = 500)
# Calculate pressure (= 50 MPa) at this density
IAPWS95("P", T = 500, rho = rho)
# Calculate dielectric constant
water.AW90(T = 500, rho = rho, P = 50)

# Density along saturation curve
T <- seq(273.15, 623.15, 25)
```

```
WP02.auxiliary(T = T)  # liquid from WP02
WP02.auxiliary("rho.vapor", T)  # vapor from WP02

# WP02.auxiliary gives a close estimate of saturation pressure...
T <- 445:455
P.sigma <- WP02.auxiliary("P.sigma", T)
# ... but alternates between being just on the liquid or vapor side
# (low rho: steam; high rho: water)
rho.IAPWS95(T, convert(P.sigma, "bar"))
# thermo()$opt$IAPWS.sat allows for choosing liquid or vapor or ""
thermo("opt$IAPWS.sat" = "")
# Shows artifactual vapor-liquid transition
water.IAPWS95("V", T, "Psat")
# The calculated Psat, while not exact, should be close enough for most
# geochemical calculations to select liquid or vapor
oldwat <- water("IAPWS95")
thermo("opt$IAPWS.sat" = "vapor")
V.vapor <- subcrt("water", T=convert(445:455, "C"))$out[[1]]$V
thermo("opt$IAPWS.sat" = "liquid") # the default
V.liquid <- subcrt("water", T=convert(445:455, "C"))$out[[1]]$V
all(V.vapor > V.liquid)  # TRUE
water(oldwat)
```

`water` *Properties of Water*

Description

Calculate thermodynamic and electrostatic properties of water.

Usage

```
water(property = NULL, T = 298.15, P = "Psat", P1 = TRUE)
water.SUPCRT92(property=NULL, T = 298.15, P = 1, P1 = TRUE)
water.IAPWS95(property=NULL, T = 298.15, P = 1)
water.DEW(property=NULL, T = 373.15, P = 1000)
```

Arguments

`property`	character, computational setting or property(s) to calculate
`T`	numeric, temperature (K)
`P`	numeric, pressure (bar), or '`Psat`' for vapor-liquid saturation
`P1`	logical, output pressure of 1 bar below 100 °C instead of calculated values of '`Psat`'?

Details

These functions compute the thermodynamic (Gibbs energy and it derivatives) and electrostatic (dielectric constant and its derivatives) properties of liquid or supercritical H_2O as a function of temperature and pressure using equations of state taken from the literature. The high-level function `water` performs different computations, depending on the setting of `thermo()$opt$water`:

'SUPCRT92' (default) or 'SUPCRT' Thermodynamic and electrostatic properties are calculated using a FORTRAN subroutine taken from the SUPCRT92 software package (Johnson et al., 1992). See more information below.

'IAPWS95' or 'IAPWS' Thermodynamic properties are calculated using an implementation in R code of the IAPWS-95 formulation (Wagner and Pruss, 2002), and electrostatic properties are calculated using the equations of Archer and Wang, 1990. See `IAPWS95` and more information below.

'DEW' Thermodynamic and electrostatic properties are calculated using the Deep Earth Water (DEW) model (Sverjensky et al., 2014). The defaults for `T` and `P` reflect the minimum values for applicability of the model; calculations at lower `T` and/or `P` points fall back to using 'SUPCRT92'. See `DEW`.

Calling the function with no arguments returns the current computational setting. Use e.g. `water("DEW")` to make the setting; the previous setting (at the time of the function call) is returned invisibly. Subsequent calculations with `water`, or other functions such as `subcrt` and `affinity`, will use that setting.

The allowed `property`s for `water` are one or more of those given below, depending on the computational setting; availability is shown by an asterisk. Note that some of the properties that can actually be calculated using the different formulations are not implemented here. Except for `rho`, the units are those used by Johnson and Norton, 1991.

Property	Description	Units	IAPWS95	SUPCRT92	DEW
`A`	Helmholtz energy	J mol^{-1}	*	*	NA
`G`	Gibbs energy	J mol^{-1}	*	*	*
`S`	Entropy	J K^{-1} mol^{-1}	*	*	NA
`U`	Internal energy	J mol^{-1}	*	*	NA
`H`	Enthalpy	J mol^{-1}	*	*	NA
`Cv`	Isochoric heat capacity	J K^{-1} mol^{-1}	*	*	NA
`Cp`	Isobaric heat capacity	J K^{-1} mol^{-1}	*	*	NA
`Speed`	Speed of sound	cm s^{-1}	NA	*	NA
`alpha`	Coefficient of isobaric expansivity	K^{-1}	NA	*	NA
`beta`	Coefficient of isothermal compressibility	bar^{-1}	NA	*	NA
`epsilon`	Dielectric constant	dimensionless	NA	*	*
`visc`	Dynamic viscosity	g cm^{-1} s^{-1}	NA	*	NA
`tcond`	Thermal conductivity	J cm^{-1} s^{-1} K^{-1}	NA	*	NA
`tdiff`	Thermal diffusivity	cm^2 s^{-1}	NA	*	NA
`Prndtl`	Prandtl number	dimensionless	NA	*	NA
`visck`	Kinematic viscosity	cm^2 s^{-1}	NA	*	NA
`albe`	Isochoric expansivity -compressibility	bar K^{-1}	NA	*	NA
`ZBorn`	Z Born function	dimensionless	NA	*	NA
`YBorn`	Y Born function	K^{-1}	*	*	NA

QBorn	Q Born function	bar^{-1}	*	*	*
daldT	Isobaric temperature derivative of expansibility	K^{-2}	NA	*	NA
XBorn	X Born function	K^{-2}	*	*	NA
NBorn	N Born function	bar^{-2}	*	NA	NA
UBorn	U Born function	bar^{-1} K^{-1}	*	NA	NA
V	Volume	cm^3 mol^{-1}	*	*	*
rho	Density	kg m^3	*	*	*
Psat	Saturation vapor pressure	bar	*	*	NA
E	Isobaric expansivity	cm^3 K^{-1}	NA	*	NA
kT	Isothermal compressibility	cm^3 bar^{-1}	NA	*	NA
de.dT	Temperature derivative of dielectric constant	K^{-1}	*	NA	NA
de.dP	Pressure derivative of dielectric constant	bar^{-1}	*	NA	NA
P	Pressure	bar	*	NA	NA
A_DH	A Debye-Huckel parameter	$kg^{0.5}$ $mol^{-0.5}$	*	*	*
B_DH	B Debye-Huckel parameter	$kg^{0.5}$ $mol^{-0.5}$ cm^{-1}	*	*	*

Call `water.SUPCRT92`, `water.IAPWS95`, or `water.DEW` with no arguments to list the available properties.

`water.SUPCRT92` interfaces to the FORTRAN subroutine taken from the SUPCRT92 package (H2O92D.F) for calculating properties of water. These calculations are based on data and equations of Levelt-Sengers et al., 1983, Haar et al., 1984, and Johnson and Norton, 1991, among others (see Johnson et al., 1992). A value of `P` set to '`Psat`' refers to one bar below 100 °C, otherwise to the vapor-liquid saturation pressure at temperatures below the critical point ('`Psat`' is not available at temperatures above the critical point). `water.SUPCRT92` provides a limited interface to the FORTRAN subroutine; some functions provided there are not made available here (e.g., using variable density instead of pressure, or calculating the properties of steam).

The stated temperature limits of validity of calculations in `water.SUPCRT92` are from the greater of 0 °C or the melting temperature at pressure, to 2250 °C (Johnson et al., 1992). Valid pressures are from the greater of zero bar or the melting pressure at temperature to 30000 bar. The present functions do not check these limits and will attempt calculations for any range of input parameters, but may return `NA` for properties that fail to be calculated at given temperatures and pressures and/or produce warnings or even errors when problems are encountered.

Starting with version 0.9-9.4, a check for minimum pressure (in valTP function in H2O92D.f) has been bypassed so that properties of H2O can be calculated using `water.SUPCRT92` at temperatures below the 0.01 °C triple point. A primary check is still enforced (Tbtm), giving a minimum valid temperature of 253.15 K.

`water.IAPWS95` is a wrapper around `IAPWS95`, `rho.IAPWS95` and `water.AW90`. `water.IAPWS95` provides for calculations at specific temperature and pressure; density, needed for `IAPWS95`, is inverted from pressure using `rho.IAPWS95`. The function also contains routines for calculating the Born functions as numerical derivatives of the static dielectric constant (from `water.AW90`). For compatibility with geochemical modeling conventions, the values of Gibbs energy, enthalpy and entropy output by `IAPWS95` are converted by `water.IAPWS95` to the triple point reference state adopted in `SUPCRT92` (Johnson and Norton, 1991; Helgeson and Kirkham, 1974).

`water.IAPWS95` also accepts setting `P` to '`Psat`', with the saturation pressure calculated from `WP02.auxiliary`; by default the returned properties are for the liquid, but this can be changed to the vapor in `thermo()$opt$IAPWS.sat`.

`A_DH` and `B_DH` are solvent parameters in the "B-dot" (extended Debye-Huckel) equation (Helgeson, 1969; Manning, 2013).

Value

A data frame, the number of rows of which corresponds to the number of input temperature-pressure pairs.

References

Archer, D. G. and Wang, P. M. (1990) The dielectric constant of water and Debye-Hückel limiting law slopes. *J. Phys. Chem. Ref. Data* **19**, 371–411. doi:10.1063/1.555853

Haar, L., Gallagher, J. S. and Kell, G. S. (1984) *NBS/NRC Steam Tables*. Hemisphere, Washington, D. C., 320 p. https://www.worldcat.org/oclc/301304139

Helgeson, H. C. and Kirkham, D. H. (1974) Theoretical prediction of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures. I. Summary of the thermodynamic/electrostatic properties of the solvent. *Am. J. Sci.* **274**, 1089–1098. doi:10.2475/ajs.274.10.1089

Helgeson, H. C. (1969) Thermodynamics of hydrothermal systems at elevated temperatures and pressures. *Am. J. Sci.* **267**, 729–804. doi:10.2475/ajs.267.7.729

Johnson, J. W. and Norton, D. (1991) Critical phenomena in hydrothermal systems: state, thermodynamic, electrostatic, and transport properties of H_2O in the critical region. *Am. J. Sci.* **291**, 541–648. doi:10.2475/ajs.291.6.541

Johnson, J. W., Oelkers, E. H. and Helgeson, H. C. (1992) SUPCRT92: A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bar and 0 to 1000°C. *Comp. Geosci.* **18**, 899–947. doi:10.1016/0098-3004(92)90029Q

Levelt-Sengers, J. M. H., Kamgarparsi, B., Balfour, F. W. and Sengers, J. V. (1983) Thermodynamic properties of steam in the critical region. *J. Phys. Chem. Ref. Data* **12**, 1–28. doi:10.1063/1.555676

Manning, C. E. (2013) Thermodynamic modeling of fluid-rock interaction at mid-crustal to upper-mantle conditions. *Rev. Mineral. Geochem.* **76**, 135–164. doi:10.2138/rmg.2013.76.5

Sverjensky, D. A., Harrison, B. and Azzolini, D. (2014) Water in the deep Earth: The dielectric constant and the solubilities of quartz and corundum to 60 kb and 1,200 °C. *Geochim. Cosmochim. Acta* **129**, 125–145. doi:10.1016/j.gca.2013.12.019

Wagner, W. and Pruss, A. (2002) The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *J. Phys. Chem. Ref. Data* **31**, 387–535. doi:10.1063/1.1461829

Examples

```
## Calculations along saturation curve
T <- seq(273.15, 623.15, 25)
# Liquid density, from SUPCRT92
water("rho", T = T, P = "Psat")
```

```
# Values of the saturation pressure, Gibbs energy
water(c("Psat", "G"), T = T, P = "Psat")
# Derivatives of the dielectric constant (Born functions)
water(c("QBorn", "YBorn", "XBorn"), T = T, P = "Psat")
# Now at constant pressure
water(c("QBorn", "YBorn", "XBorn"), T = T, P = 2000)

## Comparing the formulations
T <- convert(c(25, 100, 200, 300), "K")
# IAPWS-95
oldwat <- water("IAPWS95")
water(water.IAPWS95(), T = T)
# Deep Earth Water (DEW)
water("DEW")
water(water.DEW(), T = T, P = 1000)
# SUPCRT92 (the default)
water(oldwat)
water(water.SUPCRT92(), T = T)

## Calculating Q Born function
# After Table 22 of Johnson and Norton, 1991
T <- rep(c(375, 400, 425, 450, 475), each = 5)
P <- rep(c(250, 300, 350, 400, 450), 5)
w <- water("QBorn", T = convert(T, "K"), P = P)
# The rest is to make a neat table
w <- as.data.frame(matrix(w[[1]], nrow = 5))
colnames(w) <- T[1:5*5]
rownames(w) <- P[1:5]
print(w)
```

Index