Package 'CHNOSZ'

April 21, 2019

Date 2019-04-20

Version 1.3.2

Title Thermodynamic Calculations and Diagrams for Geochemistry

Author Jeffrey Dick [aut, cre] (<https://orcid.org/0000-0002-0687-5890>), R Core Team [ctb] (code derived from R's pmax())

Maintainer Jeffrey Dick <j3ffdick@gmail.com>

Depends R (>= 3.1.0)

Suggests limSolve, testthat, 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 has unique capabilities for comparing the compositional and thermodynamic properties of different proteins.

Encoding UTF-8

License GPL (>= 2)

BuildResaveData no

VignetteBuilder knitr

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-21 15:10:11 UTC

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.
- Be sure to check out the vignette titled *An Introduction to CHNOSZ*, which is available by following the link in `help.start` to 'User guides, package vignettes and other documentation'.
- 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`, `objective`, `revisit`, `findit`, `EOSregress`, `wjd`
- Thermodynamic data: `data`, `extdata`, `add.obigt`, `util.data`
- Thermodynamic calculations: `util.formula`, `makeup`, `util.units`, `eos`, `berman`, `nonideal`, `util.misc`
- Water properties: `water`, `util.water`, `DEW`, `IAPWS95`
- Protein properties: `protein`, `protein.info`, `add.protein`, `util.fasta`, `util.protein`, `util.seq`, `ionize.aa`, `yeast`

- Other tools: examples, eqdata, taxonomy, util.blast
- Utility functions: util.expression, util.plot, util.array, util.matrix, util.list, util.test, 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 encouragement and groudbreaking work of the late Professor Harold C. Helgeson. The revised Helgeson-Kirkham-Flowers equations of state are used in this package, together with thermodynamic properties of minerals and aqueous species from many papers coauthored by Helgeson. CHNOSZ uses Fortran code from H2O92D.f in the SUPCRT92 package (Johnson et al., 1992), with only minor modifications (masking of WRITE and STOP statements made for compatibility with the R environment and keep valTP flag TRUE to permit sub-zero °C calculations).

Work on this package at U.C. Berkeley from ca. 2003 to 2008 was supported by research grants to HCH from the U.S. National Science Foundation and Department of Energy. In 2009–2011, development of this package was based upon work supported by the National Science Foundation under grant EAR-0847616. The files in extdata/bison are derived from BLAST calculations made on the Saguaro high performance computer at Arizona State University.

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. https://doi.org/10.1016/0098-3004(92)90029-Q

---

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, E.units = "cal")
mod.obigt(...)
today()
```

Arguments

file	character, path to a file
species	character, names of species to load from file
force	logical, force replacement of already existing species?
E.units	character, units of energy, 'cal' or 'J'
...	character or numeric, properties of species to modify in the thermodynamic database

Details

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.

file is first matched against the names of files in the extdata/OBIGT directory packaged with CHNOSZ. In this case, the filename suffixes are removed, so 'DEW_aq', 'organic_aq', and 'organic_cr' are valid names. The function also accepts single matches with the state suffix dropped ('DEW' but not 'organic'). 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).

Given the default setting of E.units, the function does not perform any unit conversions. If E.units is set to 'J', then the thermodynamic parameters are converted from units of Joules to calories, as used in the CHNOSZ database.

When adding 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. These changes are lost if you reload the database by calling data(thermo) or if you quit the R session without saving it. The name of the species to add or change must be supplied as the first argument of ... or as a named argument (named 'name'). 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 results if the formula is not valid (i.e. can not be parsed bymakeup). Additional arguments refer to the name of the property(s) to be updated and are matched to any part of compound column names in thermo$obigt, such as 'z' or 'T' in 'z.T'. Unless

'state' is specified as one of the properties, its value is taken from thermo()optstate. When adding species, properties that are not specified become NA, except for 'state', which takes a default value from thermo()optstate, and 'z.T', which for aqueous species is set to the charge calculated from the chemical formula (otherwise, NA charge for newly added species would trigger the AkDi model). The values provided should be in the units specifed in the documentation for the thermo data object, including any order-of-magnitude scaling factors.

today returns the current date in the format adopted for thermo()$obigt (inherited from SUPCRT-format data files) e.g. '13.May.12' for May 13, 2012.

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. https://doi.org/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. https://doi.org/10.1007/s00239-014-9616-1

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. https://doi.org/10.1016/0016-7037(89)90341-4

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. https://doi.org/10.1016/S0009-2541(00)00263-1

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. https://doi.org/10.1016/S0016-7037(97)00009-4

See Also

thermo, util.data, mod.buffer

Examples

```
## modify an existing species (example only)
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 a species
iCl2O <- mod.obigt("Cl2O", G=20970)
info(iCl2O)
# add a species with a name that is different from the formula
mod.obigt("buckminsterfullerene", formula="C60", state="cr", date=today())
# retrieve the species data (thermodynamic properties in this toy example are NA)
info(info("C60"))
# 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
})
# reset default database
obigt()
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))
}))

# another use of add.obigt()
# compare Delta G of AABB = UPBB + H2O
# (Figure 9 of Kitadai, 2014)
E.units("J")
# default database has values from Kitadai, 2014
Kit14 <- subcrt(c("[AABB]", "[UPBB]", "H2O"), c(-1, 1, 1), T = seq(0, 300, 10))
# optional file OldAA has superseded values of [UPBB] from Dick et al., 2006
add.obigt("OldAA")
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")
```

```
# reset database *and* settings (units)
reset()

# Another use of add.obigt(): calculate Delta G of
# H4SiO4 = SiO2 + 2H2O using different 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(-100, 600))
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", legend = as.expression(stxt), bty = "n")
legend("topright", c("Stef\u00e1nsson, 2001", "Shock et al., 1989",
  "Apps and Spycher, 2004"), lty = c(0, 1, 2), bty = "n")
reset()
```

`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)
seq2aa(protein, sequence)
aasum(aa, abundance = 1, average = FALSE, protein = NULL, organism = NULL)
```

Arguments

aa	data frame, amino acid composition in the format of `thermo()$protein`
protein	character, name of protein; numeric, indices of proteins (rownumbers of `thermo$protein`)
sequence	character, protein sequence
...	additional arguments passed to `read.csv`
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, in the format of thermo()$protein, corresponding to the provided sequence. Here, the protein argument indicates the name of the protein with an underscore (e.g. 'LYSC_CHICK').

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.

Given amino acid compositions returned by the *aa functions described above, 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. The value returned by this function is the rownumbers of thermo()$protein that are added and/or replaced.

See Also

read.fasta, uniprot.aa, yeast.aa for other ways of getting amino acid compositions.

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

protein for examples of affinity calculations and diagrams.

Examples

```
# manually adding a new protein
# Human Gastric juice peptide 1
aa <- seq2aa("GAJU_HUMAN", "LAAGKVEDSD")
ip <- add.protein(aa)
stopifnot(protein.length(ip)==10)
# the chemical formula of this peptide
as.chemical.formula(protein.formula(ip)) # "C41H69N11O18"
# we can also calculate a formula without using add.protein
aa <- seq2aa("pentapeptide_test", "ANLSG")
as.chemical.formula(protein.formula(aa))
```

---

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 = -3)
```

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

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 is $A=RT\ln(K/Q)$, where K denotes the equilibrium constant of the reaction and Q stands for the activity product of the species in the reaction. 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.

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 minimum and maximum values of variable identified by the name of the argument, and res denotes the resolution, or number of points along which to do the calculations; res is assigned a default value of 128 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.

The function can also be used 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.

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 chemical 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. https://doi.org/10.1016/j.gca.2008.03.004

See Also

ionize.aa, activated if proteins are among the species of interest and 'H+' is in the basis. equilibrate for using the results of affinity to calculate equilibrium activities of species, and diagram to plot the results.

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
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,
# user units (default: cal/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)
```

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 species definition?

Details

The basis species represent the possible range of chemical compositions for all the species of interest. Any valid set of basis species used here must meet 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 first argument. Alternatively, the first argument may consist of numeric values indicating the species indices (rownumbers in `thermo$obigt`), but a mixture of character and numeric values will generate an error. The special names '`pH`', '`pe`' and '`Eh`' can be included in the `species` argument; they get translated into the names of the proton ('`H+`') and electron ('`e-`') as appropriate.

The physical states or logarithms of activities of species in the basis definition can be changed by calling `basis` with the formulas of species that are in the basis set, or their species indices. If either of the second or third arguments to `basis` is of type character, it refers to the name of a state (if present in `thermo()$obigt$state`) or to the name of 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.

Whenever `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. If 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^+

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. https://doi.org/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 https://doi.org/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
# one, two and three element examples
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, thisinfo = NULL, check.G = FALSE,
       calc.transition = TRUE, calc.disorder = TRUE, units = "cal")
```

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)
thisinfo	dataframe, row for mineral from thermo()$obigt
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?
units	character, energy units, 'cal' or 'J'

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 (if available) named 'berman.csv' in the working directory.

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()$obigt)
GfPrTr	standard Gibbs energy at 298.15 K and 1 bar (J mol^{-1}) (Benson-Helgeson convention)
HfPrTr	standard enthalpy at 298.15 K and 1 bar (J mol^{-1})
SPrTr	standard entropy at 298.15 K and 1 bar (J mol^{-1} K^{-1})
VPrTr	standard volume at 298.15 K and 1 bar (J mol^{-1})
k0 ... k6	k0 (J mol^{-1} K^{-1}) to k6
v1	v1 (K^{-1}) * 10^5
v2	v2 (K^{-2}) * 10^5
v3	v3 (bar^{-1}) * 10^5
v4	v4 (bar^{-2}) * 10^8
Tlambda	T_λ (K)
Tref	T_{ref} (K)
dTdP	dT / dP (K bar^{-1})
l1	l1 ((J/mol)$^{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.

Providing thisinfo avoids searching for the mineral in thermo()$obigt, potentially saving some running time. 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 expressed in the given units ('cal' or 'J'), and V (cm^3 mol^{-1}).

References

Anderson, G. M. (2005) *Thermodynamics of Natural Systems*, 2nd ed., Cambridge University Press, 648 p. http://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. https://doi.org/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. https://doi.org/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. https://doi.org/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. http://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
stopifnot(all(abs(DGrxn) < 100))

### compare mineral stabilities in the Berman and Helgeson datasets
### on a T - log(K+/H+) diagram, after Sverjensky et al., 1991
### (doi:10.1016/0016-7037(91)90157-Z)
## set up the system: basis species
basis(c("K+", "Al+3", "quartz", "H2O", "O2", "H+"))
# use pH = 0 so that aK+ = aK+/aH+
basis("pH", 0)
# load the species
species(c("K-feldspar", "muscovite", "kaolinite",
          "pyrophyllite", "andalusite"), "cr")
## start with the data from Helgeson et al., 1978
add.obigt("SUPCRT92")
# calculate affinities in aK+ - temperature space
# exceed.Tr: enable calculations above stated temperature limit of pyrophyllite
res <- 400
a <- affinity(`K+` = c(0, 5, res), T = c(200, 650, res), P = 1000, exceed.Ttr = TRUE)
# make base plot with colors and no lines
diagram(a, xlab = ratlab("K+", molality = TRUE), lty = 0, fill = "terrain")
# add the lines, extending into the low-density region (exceed.rhomin = TRUE)
```

```
a <- affinity(`K+` = c(0, 5, res), T = c(200, 650, res), P = 1000,
              exceed.Ttr = TRUE, exceed.rhomin = TRUE)
diagram(a, add = TRUE, names = NULL, col = "red", lwd = 1.5)
# the list of references:
ref1 <- thermo.refs(species()$ispecies)$key
## now use the (default) data from Berman, 1988
# this resets the thermodynamic database
# without affecting the basis and species settings
obigt()
# we can check that we have Berman's quartz
# and not coesite or some other phase of SiO2
iSiO2 <- rownames(basis()) == "SiO2"
stopifnot(info(basis()$ispecies[iSiO2])$name == "quartz")
# Berman's dataset doesn't have the upper temperature limits,
# so we don't need exceed.Ttr here
a <- affinity(`K+` = c(0, 5, res), T = c(200, 650, res), P = 1000, exceed.rhomin = TRUE)
diagram(a, add = TRUE, names = NULL, col = "blue", lwd = 1.5)
# the list of references:
ref2 <- thermo.refs(species()$ispecies)$key
ref2 <- paste(ref2, collapse = ", ")
# add legend and title
legend("top", "low-density region", text.font = 3, bty = "n")
legend("topleft", describe.property(c("P", "IS"), c(1000, 1)), bty = "n")
legend("left", c(ref1, ref2),
       lty = c(1, 1), lwd = 1.5, col = c(2, 4), bty = "n")
title(main = syslab(c("K2O", "Al2O3", "SiO2", "H2O", "HCl")), line = 1.8)
title(main = "Helgeson and Berman minerals, after Sverjensky et al., 1991",
      line = 0.3, font.main = 1)
# cleanup for next example
reset()

# 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
E.units("J")
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
stopifnot((Gex_BA96 - Gex_Ber88)[1] > 0)
# the curves cross at about 725 deg C (BA96 Fig. 8)
# (actually, in our calculation they cross closer to 800 deg C)
```

```
stopifnot(T[which.min(abs(Gex_BA96 - Gex_Ber88))] == 800)
# reset the database (thermo()$opt$E.units, 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 = thermo()$opt$state,
  logact = -3)
```

Arguments

name	character, name of buffer to add to or find in thermo()$buffers.
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. It is possible to set the logarithms of activities of the species in the buffer through the logact argument; if this is missing unit activity is assigned to crystalline species

in buffer, otherwise (for aqueous species) the default value of activity is 10^{-3}. 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. http://www.worldcat.org/oclc/552690

See Also

`protein` for an example using a buffer made of proteins.

Examples

```
## list the buffers
thermo()$buffers
# 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
(logaCO2 <- affinity(return.buffer=TRUE)$CO2)
stopifnot(all.equal(logaCO2, -7.05752136))
# 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)
# see below for a different strategy using the
# 'type' argument of diagram

## 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)$values
# setup H2S and O2 to be buffered by PPM
basis(c("H2S","O2"),c("PPM","PPM"))
# inspect values of H2S activity and O2 fugacity
affinity(T=c(200, 400, 11), P=2000, return.buffer=TRUE, exceed.Ttr=TRUE)
# now, the affinities of formation reactions of
# species in the buffer are all equal to zero
print(a <- affinity(T=c(200, 400, 11), P=2000,
  exceed.Ttr=TRUE)$values)
for(i in 1:length(a)) stopifnot(isTRUE(
  all.equal(as.numeric(a[[i]]),rep(0,length(a[[i]])))))

## 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],sep=""))
}
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)

# one can solve for the logarithm of activity of a
```

```
# basis species using the 'type' argument of diagram
basis("CHNOS")
basis("CO2", 999)
species("acetic acid", -3)
a <- affinity(O2=c(-85, -70, 4), T=c(25, 100, 4))
# write a title with formulas and subscripts
lCO2 <- axis.label("CO2")
main <- substitute(a~~b~~c,list(a=lCO2, b="buffered by",
  c="acetic acid"))
d <- diagram(a, type="CO2", main=main)
species(1, -10)
a <- affinity(O2=c(-85, -70, 4), T=c(25, 100, 4))
d <- diagram(a, type="CO2", add=TRUE, lty=2)
# add a legend
lAC <- expr.species("CH3COOH", log=TRUE)
ltext <- c(as.expression(lAC), -3, -10)
lty <- c(NA, 1, 2)
legend("topright", legend=ltext, lty=lty, bg="white")
# do return.buffer and diagram(type=...) give the same results?
and <- as.numeric(d$plotvals[[1]])
basis("CO2", "AC")
mod.buffer("AC", logact=-10)
a.buffer <- affinity(O2=c(-85, -70, 4), T=c(25, 100, 4),
  return.buffer=TRUE)
ana <- as.numeric(unlist(a.buffer[[1]]))
stopifnot(all.equal(ana, and))
```

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://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. https://doi.org/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. https://doi.org/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 speecies, 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, lwd = par("lwd"), dotted = NULL, spline.method = NULL,
  contour.method = "edge",
  # colors
  col = par("col"), col.names = par("col"), fill = NULL,
  fill.NA = "gray80", limit.water = TRUE,
  # field and line labels
  names = NULL, format.names = TRUE, bold = FALSE, italic = FALSE,
  font = par("font"), family = par("family"), adj = 0.5, dy = 0, srt = 0,
  # title and legend
  main = NULL, legend.x = NA,
  # plotting controls
  add = FALSE, plot.it = TRUE, tplot = TRUE, ...)
strip(affinity, ispecies = NULL, col = NULL, ns = NULL,
  xticks = NULL, ymin = -0.2, xpad = 1, cex.names = 0.7)
find.tp(x)
```

Arguments

eout	list, object returned by equilibrate or affinity
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 (rescale 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
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).
col	character, color of activity lines (1D diagram) or predominance field boundaries (2D diagram), or colors of bars in a strip diagram (strip)
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	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
dy	numeric, y offset for line labels
srt	numeric, rotation for line labels
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?
affinity	list, object returned by affinity
ispecies	numeric, which species to consider (default of NULL is to consider all species)
ns	numeric, numbers of species, used to make inset plots for strip diagrams

xticks	numeric, location of supplemental tick marks on x-axis
ymin	numeric, lower limit of y-axis
xpad	numeric, amount to extend x-axis on each side
x	matrix, value of the predominant list element from diagram
...	additional arguments passed to plot or barplot

Details

diagram takes as its primary input the results from equilibrate and displays diagrams representing the equilibrium chemical activities of the species. 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 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 formuls unless format.names is FALSE. Use the names argument to override the default species names. Alternatively, supply a numeric value to names to indicate the 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.

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.

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` produces transparent predominance fields. `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 for which NA values are present, possibly by using `equilibrate` at extreme conditions (see `test-diagram.Rd`). `fill.NA` is also used to specify the color outside the water stability limits on Eh-pH or pe-pH diagrams, when `limit.water` is TRUE. Note that the default for `fill.NA` is automatically changed to '`transparent`' when `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. To go back to the old behavior for drawing lines, set `dotted` to '`0`'. The old behavior does not follow `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.

`normalize` and `as.residue` apply only to the 2-D diagrams, and only when `eout` is the output from `affinity`. With `normalize`, the activity boundaries are calculated as between the residues of the species (the species divided by the balance coefficients), then the activities are rescaled to the whole species formulas. With `as.residue`, the activity boundaries are calculated as between the residues of the species, and no rescaling is performed.

Affinity Diagrams

The function behaves differently when the output from `affinity` is being used instead of the equilibrium activities from `equilibrate`. If `type` is '`auto`', and the number of dimensions is 0 or 1, the property computed by `affinity` for each species is plotted. This is usually the affinity of the formation reaction, but can be set to some other property, such as the equilibrium constant ('`logK`'). If `type` is '`auto`', and the number of dimensions is 2, then equilibrium predominance (maximum affinity) fields are plotted. This algorithm is based on a comparison of the affinities of the formation reactions scaled by the balancing coefficients that are determined 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 is beyond the saturation line, the function prints a message instead. If `type` is the name of a basis species, 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, using the method specified in `contour.method`. The '`saturation`' diagram can handle multiple species, but if `type` is the name a basis species, then only the first species of interest is used in the calculation, and a warning is produced if there is more than one.

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 tests/testthat/test-logmolality.R). 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).

Strip Diagrams

A different incarnation of 1-D speciation diagrams is provided by strip. This function generates any number of strip diagrams in a single plot. The diagrams are made up of colors bars whose heights represent the relative abundances of species; the color bars are arranged in order of abundance and the total height of the stack of colors bars is constant. If ispecies is a list, the number of strip diagrams is equal to the number of elements of the list, and the elements of this list are numeric vectors that identify the species to consider for each diagram. The strips are labeled with the names of ispecies. If col is NULL, the colors of the bars are generated using rainbow. Supplemental ticks can be added to the x-axis at the locations specified in xtick; they are larger than the standard ticks and have colors corresponding to those of the color bars. ymin can be decreased in order to add more space at the bottom of the plot, and xpad can be changed in order to increase or decrease the size of the x-axis relative to the width of the strips. An inset dot-and-line plot is created below each strip if ns is given. This argument has the same format as ispecies, and can be used e.g. to display the relative numbers of species for comparison with the stability calculations.

Value

The function returns an invisible list containing, first, the contents of eout, i.e. the provided output of affinity or equilibrate. To this are added the name of the plotted variable in plotvar, the plotted values in plotvals, and the names used for labeling the plot in names. 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, the output also contains predominant, giving the numbers (from the species definition) of the predominant (aka maximum-affinity) species at each grid point. The rows and columns of predominant correspond to the x- and y-variables, respectively. Finally, the output for 2-D diagrams contains a lines component, giving the x- and y-coordinates of the field boundaries computed using contourLines; the values are padded to equal length with NAs to faciliate exporting the results using write.csv.

References

Aksu, S. and Doyle, F. M. (2001) Electrochemistry of copper in aqueous glycine solutions. *J. Electrochem. Soc.* **148**, B51–B57. https://doi.org/10.1149/1.1344532

Helgeson, H. C. (1970) A chemical and thermodynamic model of ore deposition in hydrothermal systems. *Mineral. Soc. Amer. Spec. Pap.* **3**, 155–186. http://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. http://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. https://doi.org/10.1111/j.1472-4669.2007.00099.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. https://doi.org/10.1127/0935-1221/2006/0018-0175

Tagirov, B. and Schott, J. (2001) Aluminum speciation in crustal fluids revisited. *Geochim. Cosmochim. Acta* **65**, 3965–3992. https://doi.org/10.1016/S0016-7037(01)00705-0

See Also

Other examples are present in the help for protein and buffer, and even more can be found in demos. See the vignette *Hot-spring proteins in CHNOSZ* for an example of the `strip` charts.

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", "O2", "S2"))
species(c("iron", "ferrous-oxide", "magnetite",
  "hematite", "pyrite", "pyrrhotite"))
# the calculations include the phase 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")
# the first four species show up in order near pe=15
stopifnot(all.equal(unique(d$predominant[, 183]), 1:4))
water.lines(d, lwd=2)
text(3, 22, describe.basis(thermo()$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
# this uses a HCL palette for the fill colors
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$vals[[2]])[tp[1, 1]]
points(Ttp, Ptp, pch=10, cex=5)
# some testing of the overall geometry
stopifnot(species()$name[d$predominant[1, 1]]=="andalusite")
stopifnot(species()$name[d$predominant[1, 101]]=="kyanite")
stopifnot(species()$name[d$predominant[99, 101]]=="sillimanite")
```

---

eos *Equations of State*

Description

Calculate thermodynamic properties using the revised Helgeson-Kirkham-Flowers (HKF) or Akinfiev-Diamond (AkDi) equations of state for aqueous species, or using a generic heat capacity equation for crystalline, gas, and liquid species.

Usage

```
cgl(property = NULL, parameters = NULL, T = 298.15, P = 1)
hkf(property = NULL, parameters = NULL, T = 298.15, P = 1,
  contrib = c("n", "s", "o"), H2O.props = "rho")
AkDi(property = NULL, parameters = NULL, T = 298.15, P = 1, isPsat = TRUE)
```

Arguments

property	character, name(s) of properties to calculate
parameters	dataframe, species parameters as one or more rows from thermo()$obigt
T	numeric, temperature(s) at which to calculate properties (K)
P	numeric, pressure(s) at which to calculate properties (bar)
contrib	character, which contributions to consider in the revised HKF equations equations of state: (n)onsolvation, (s)olvation (the ω terms), or (o)rigination contributions (i.e., the property itself at 25 °C and 1 bar). Default is c("n","s","o"), for all contributions
H2O.props	character, properties to calculate for water
isPsat	logical, is this a calculation along the liquid-vapor saturation curve (Psat)?

Details

The equations of state permit the calculation of the standard molal properties of species as a function of temperature and pressure. The `property` argument is required and refers to one or more of '`G`', '`H`', '`S`', '`Cp`' and '`V`', and for aqueous species only, '`kT`' and '`E`'. The units of these properties are the first ones shown in the description for `subcrt`. The names of the properties are matched without regard to case.

`hkf` implements the revised HKF equations of state (Helgeson et al., 1981; Tanger and Helgeson, 1988; Shock and Helgeson, 1988). The equations-of-state parameters are `a1`, `a2`, `a3`, `a4`, `c1`, `c2`, `omega` and `Z`; the units of these parameters are as indicated for `thermo$obigt`, without the order of magnitude multipliers. Note that the equation-of-state parameter `Z` (appearing in the g-function for the temperature derivatives of the omega parameter; Shock et al., 1992) is taken from `thermo()$obigt` and *not* from the `makeup` of the species. `H2O.props` is an optional argument that lists the properties of water that should be returned; it is used by `subcrt` so that the time-consuming water calculations are only performed once.

The temperature and pressure derivatives of the `omega` parameter for charged species (where `Z != 0`, but not for the aqueous proton, H+) are calculated using the g- and f-functions described by Shock et al., 1992 and Johnson et al., 1992. If the IAPWS-95 or DEW equations are activated (see `water`), only the g-function (applicable to '`G`'), but not its derivatives (needed for '`H`', '`S`', '`Cp`', and '`V`'), is calculated.

The parameters in the `cgl` equations of state for crystalline, gas and liquid species (except liquid water) include `V`, `a`, `b`, `c`, `d`, `e`, `f` and `lambda`. The terms denoted by `a`, `b` and `c` correspond to the Maier-Kelley equation for heat capacity (Maier and Kelley, 1932); the additional terms are useful for representing heat capacities of minerals (Robie and Hemingway, 1995) and gaseous or liquid organic species (Helgeson et al., 1998). The standard molal volumes ('`V`') of species in these calculations are taken to be independent of temperature and pressure.

For both `hkf` and `cgl`, if at least one equations-of-state parameter for a species is provided, any NA values of the other parameters are reset to zero. If all equations-of-state parameters are NA, but values of '`Cp`' and/or '`V`' are available, those values are used in the integration of '`G`', '`H`' and '`S`' as a function of temperature.

`AkDi` provides the Akinfiev-Diamond model for aqueous species (Akinfiev and Diamond, 2003). To run this code, the database must also include the corresponding gasesous species (with the same name or chemical formula). Currently, only the standard chemical potential (Gibbs energy) is calculated.

Value

A list of length equal to the number of species (i.e., number rows of `parameters`). Each element of the list contains a dataframe, each column of which corresponds to one of the specified properties; the number of rows is equal to the number of pressure-temperature points. Furthermore, in `hkf`, the output is a list consisting of the above-described object (named '`aq`') and a data frame of the calculated properties of water (named '`H2O`').

Warning

The range of applicability of the revised HKF equations of state for aqueous species corresponds to the stability region of liquid water or the supercritical fluid with density greater than 0.35 g/cm3, and between 0 to 1000 °C and 1 to 5000 bar (Tanger and Helgeson, 1988; Shock and Helgeson,

1988). The hkf function does not check these limits and will compute properties as long as the requisite electrostatic properties of water are available. There are conceptually no temperature limits (other than 0 Kelvin) for the validity of the cgl equations of state. However, the actual working upper temperature limits correspond to the temperatures of phase transitions of minerals or to those temperatures beyond which extrapolations from experimental data become highly uncertain. These temperature limits are stored in the thermodynamic database for some minerals, but cgl ignores them; however, subcrt warns if they are exceeded.

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. https://doi.org/10.1016/S0016-7037(02)01141-9

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. https://doi.org/10.2475/ajs.281.10.1249

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. https://doi.org/10.1016/S0016-7037(97)00219-6

Maier, C. G. and Kelley, K. K. (1932) An equation for the representation of high-temperature heat content data. *J. Am. Chem. Soc.* **54**, 3243–3246. https://doi.org/10.1021/ja01347a029

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. http://www.worldcat.org/oclc/32590140

Shock, E. L. and Helgeson, H. C. (1988) Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Correlation algorithms for ionic species and equation of state predictions to 5 kb and 1000°C. *Geochim. Cosmochim. Acta* **52**, 2009–2036. https://doi.org/10.1016/0016-7037(88)90181-0

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. https://doi.org/10.1039/FT9928800803

Tanger, J. C. IV and Helgeson, H. C. (1988) Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Revised equations of state for the standard partial molal properties of ions and electrolytes. *Am. J. Sci.* **288**, 19–98. https://doi.org/10.2475/ajs.288.1.19

See Also

info for retrieving equations of state parameters from the thermodynamic database, water for equations of state of water, subcrt for interactive use of these equations.

Examples

```
## aqueous species
CH4aq <- info(info("methane", "aq"))
hkf(property = "Cp", parameters = CH4aq)
# the non-solvation heat capacity
hkf(property = "Cp", parameters = CH4aq, contrib = "n")
# at different temperature and pressure
hkf(property = "Cp", parameters = CH4aq, T = c(373.15,473.15), P = 1000)

## crystalline, gas, liquid species
CH4gas <- info(info("methane", "gas"))
cgl(property = "Cp", parameters = CH4gas)
# melting and vaporization of octane
C8H18par <- info(info(rep("octane", 3), c("cr", "liq", "gas")))
myT <- seq(200, 420, 10)
DG0f <- cgl(property = "G", parameters = C8H18par, T = myT, P = 1)
cbind(T = myT, which.pmax(DG0f, pmin = TRUE))  # 1 = cr, 2 = liq, 3 = gas
# compare that result with the tabulated transition temperatures
print(C8H18par)
```

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 $Cp^\circ = c_1 + c_2/(T - \Theta)^2 + \omega T X$. 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. https://doi.org/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. https://doi.org/10.1016/S0016-7037(01)00717-7

See Also

The vignette *Regressing thermodynamic data* 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/Cp.CH4.HW97.csv", package="CHNOSZ")
d <- read.csv(f)
# have to 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/V.CH4.HWM96.csv", package="CHNOSZ")
d <- read.csv(f)
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
```

`eqdata` *Read data from an EQ6 output file*

Description

Extract computational results for aqueous species, solid phases, mineral saturation states, or speciation summaries at each step of reaction progress in an EQ6 output file. The results are written to a comma-separated value file that can be read by other programs. The function has been tested with output files generated by EQ3/6 version 7.1 running on a Unix platform. Currently there is only partial support for version 8.0a (reading data from aqueous species blocks).

Usage

```
eqdata(file, species, property = "log act", outfile = TRUE)
```

Arguments

`file`	character, path to EQ6 output file
`species`	character, name(s) of species or minerals
`property`	character, property to get
`outfile`	logical or character, file for saving results

Details

The first argument, `file`, is the name of the EQ6 (Wolery, 1992; Wolery and Daveler, 1992) output file. `species` indicates the aqueous species, solid phases, minerals, or basis species for which you want values; multiple names can be provided except for basis species, which can be a single value. `property` indicates the property to retrieve. Specifying a value other than one listed below will cause an error.

- Aqueous species: '`conc`', '`log conc`', '`log g`', or '`log act`'
- Solid phases: '`log moles`', '`moles`', '`grams`', or '`volume, cc`'
- Minerals (saturation states): '`affinity, kcal`'
- Basis species (speciation): '`molal conc`' or '`per cent`'

The result of the function is a data frame (returned invisibly), with columns `zi` (reaction progress), `T` (temperature in °C, `aH2O` (activity of water) and one column for each of the requested `species` or, for speciation of basis species, one column for each unique species found in all of the speciation summary blocks for that basis species. Values are listed as NA (not available) for species or phases that are not present in the EQ6 output at any of the increments of reaction progress.

If `outfile` is TRUE, the result is saved in a file named like '`file`'.'`property`'.csv, in the same directory as `file`. The name of the `outfile` can be provided to override this naming scheme, or this argument can be set to FALSE or NULL, to turn off writing the result to a file.

Thanks to Peter Canovas and Everett Shock for helping to test the code and offering ideas for improvements.

References

Wolery, T. J. (1992) EQ3/6, A Software Package for Geochemical Modeling of Aqueous Systems: Package Overview and Installation Guide (Version 7.0). Lawrence Livermore National Laboratory, UCRL-MA-110662 PT I. http://www.wipp.energy.gov/library/cra/2009_cra/references/Others/Wolery_1992_EQ36_A_Software_Package_for_Geochemical_Modeling_of_Aqueous_Systems_ERMS241375.pdf

Wolery, T. J. and Daveler, S. A. (1992) EQ6, A Computer Program for Reaction Path Modeling of Aqueous Geochemical Systems: Theoretical Manual, User's Guide, and Related Documentation (Version 7.0). Lawrence Livermore National Laboratory, UCRL-MA-110662 PT IV. http://www.wipp.energy.gov/library/cra/2009_cra/references/Others/Wolery_Daveler_1992_EQ36_A_Computer_Program_for_Reaction_Path_Modeling_of_Aqueous_Geochemical_Systems_ERMS241379.pdf

Examples

```
## Not run:
# if an EQ6 output file named "rainbow2.6o" is in the current
# working directory, the following command will output values
# of log act (logarithm of activity) for the selected aqueous
# species to a file named rainbow2.6o.log act.csv
eqdata("rainbow2.6o",c("h+","sio2,aq","h2,aq"),"log act")
## End(Not run)
```

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=1:length(aout$values), normalize=FALSE, as.residue=FALSE,
  method=c("boltzmann", "reaction"), tol=.Machine$double.eps^0.25)
equil.boltzmann(Astar, n.balance, loga.balance)
equil.reaction(Astar, n.balance, loga.balance, tol=.Machine$double.eps^0.25)
```

Arguments

aout	list, output from affinity
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

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, which may be calculated from a multidimensional grid of conditions. 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.

aout contains the chemical affinities of formation reactions of each species of interest. 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 calculation.

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. https://doi.org/10.1186/1467-4866-9-10

See Also

`diagram` has examples of using `equilibrate` to make equilibrium activity diagrams. `revisit` can be used to perform further analysis of the equilibrium activities. `palply` is used by both `equil.reaction` and `equil.boltzmann` to parallelize intensive parts of the calculations.

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))
# probably close enough for most purposes
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", "revisit", "findit", "ionize", "buffer",
  "protbuff", "yeastgfp", "glycinate", "mosaic", "copper",
  "arsenic", "solubility", "gold", "contour", "wjd", "bugstab",
  "Shh", "saturation", "adenine", "DEW", "lambda", "TCA", "aluminum",
  "bison", "AkDi"),
  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 `demo`s 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
`revisit`	Coefficient of variation of metastable equilibrium activities of proteins
`findit`	Minimize the standard deviation of logarithms of activities of sulfur species
`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
yeastgfp	Subcellular locations: $\log f_{O_2}$ - $\log a_{H_2O}$ and $\log a$ - $\log f_{O_2}$ diagrams (Dick, 2009)
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)
wjd	G minimization: prebiological atmospheres (Dayhoff et al., 1964) and cell periphery of yeast
dehydration	$\log K$ of dehydration reactions; SVG file contains tooltips and links
bugstab	Formation potential of microbial proteins in colorectal cancer (Dick, 2016)
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)
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
bison	Average oxidation state of carbon in proteins for phyla at Bison Pool (Dick and Shock, 2013)
AkDi	Henry's constant of dissolved gases (Akinfiev and Diamond, 2003)

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. https://doi.org/10.1016/S0016-7037(02)01141-9

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. http://pleiades.online/cgi-perl/search.pl?type=abstract&name=geochem&number=10&year=1&page=990

Aksu, S. and Doyle, F. M. (2001) Electrochemistry of copper in aqueous glycine solutions. *J. Electrochem. Soc.* **148**, B51–B57. https://doi.org/10.1149/1.1344532

Amend, J. P. and Shock, E. L. (1998) Energetics of amino acid synthesis in hydrothermal ecosystems. *Science* **281**, 1659–1662. https://doi.org/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. https://doi.org/10.1016/S0168-6445(00)00062-0

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. https://doi.org/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. https://doi.org/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. http://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. https://doi.org/10.1016/j.gca.2016.08.028

Dayhoff, M. O. and Lippincott, E. R. and Eck, R. V. (1964) Thermodynamic Equilibria In Prebiological Atmospheres. *Science* **146**, 1461–1464. https://doi.org/10.1126/science.146.3650.1461

Dick, J. M. (2009) Calculation of the relative metastabilities of proteins in subcellular compartments of *Saccharomyces cerevisiae*. *BMC Syst. Biol.* **3**:75. https://doi.org/10.1186/1752-0509-3-75

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. https://doi.org/10.1371/journal.pone.0022782

Dick, J. M. and Shock, E. L. (2013) A metastable equilibrium model for the relative abundance of microbial phyla in a hot spring. *PLoS ONE* **8**, e72395. https://doi.org/10.1371/journal.pone.0072395

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

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

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

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. https://doi.org/10.1016/0098-3004(92)90029-Q

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. https://doi.org/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. https://doi.org/10.1007/s12665-010-0652-x

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. https://doi.org/10.2138/rmg.2013.75.5

Schulte, M. D. and Shock, E. L. (1995) Thermodynamics of Strecker synthesis in hydrothermal systems. *Orig. Life Evol. Biosph.* **25**, 161–173. https://doi.org/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. https://doi.org/10.1016/0016-7037(95)00058-8

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. https://doi.org/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. https://doi.org/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. http://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. https://doi.org/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. https://doi.org/10.1016/0016-7037(91)90157-Z

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. https://doi.org/10.1038/ngeo2291

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. https://doi.org/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. https://doi.org/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. https://doi.org/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 in other data files, the names of sanidine and microcline were changed to K-feldspar,high and K-feldspar,low.
- `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 `abundance` contain protein abundance and microbial occurrence data:

- `TBD+05.csv` lists genes with transcriptomic expression changes in carbon limitation stress response experiments in yeast (Tai et al., 2005). See `yeast.aa` for an example that uses this file.
- `yeastgfp.csv.xz` Has 28 columns; the names of the first five are `yORF`, `gene name`, `GFP tagged?`, `GFP visualized?`, and `abundance`. The remaining columns correspond to the 23 subcellular localizations considered in the YeastGFP project (Huh et al., 2003 and Ghaemmaghami et al., 2003) and hold values of either `T` or `F` for each protein. 'yeastgfp.csv' was downloaded on 2007-02-01 from http://yeastgfp.ucsf.edu using the Advanced Search, setting options to download the entire dataset and to include localization table and abundance, sorted by orf number. See `yeastgfp` and `demo("yeastgfp")` for examples that use this file.
- `microbes.csv` has data for microbial occurrence (i.e. relative enrichement) in colorectal cancer and normal tissue. The file is from the Supporting Information of Dick (2016). This file is used by `demo("bugstab")`.

Files in bison contain BLAST results and taxonomic information for an environmental metagenome from the Bison Pool hot spring in Yellowstone National Park:

- bisonN_vs_refseq57.blast.xz, bisonS..., bisonR..., bisonQ..., bisonP... are partial tabular BLAST results for proteins in the Bison Pool Environmental Genome. Protein sequences predicted in the metagenome were downloaded from the Joint Genome Institute's IMG/M system on 2009-05-13. The target database for the searches was constructed from microbial protein sequences in National Center for Biotechnology Information (NCBI) RefSeq database version 57, representing 7415 microbial genomes. The 'blastall' command was used with the default setting for E value cuttoff (10.0) and options to make a tabular output file consisting of the top 20 hits for each query sequence. The function read.blast was used to extract only those hits with E values less than or equal to 1e-5 and with sequence similarity (percent identity) at least 30 percent, and to keep only the first hit for each query sequence. The function write.blast was used to save partial BLAST files (only selected columns). The files provided with CHNOSZ contain the first 5,000 hits for each sampling site at Bison Pool, representing between about 7 to 15 percent of the first BLAST hits after similarity and E value filtering.
- gi.taxid.txt.xz is a table that lists the sequence identifiers (gi numbers) that appear in the example BLAST files (see above), together with the corresponding taxon ids used in the NCBI databases. This file is *not* a subset of the complete 'gi_taxid_prot.dmp.gz' available at ftp://ftp.ncbi.nih.gov/pub/taxonomy/ but instead is a subset of 'gi.taxid.txt' generated from the RefSeq release catalog using 'gencat.sh' in the refseq directory. See id.blast for an example that uses this file and the BLAST files described above.

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.Rmd 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.
- RT71.csv pH titration measurements for unfolded lysozyme ('LYSC_CHICK') taken from Roxby and Tanford, 1971. pH is in the first column, net charge in the second. See ionize.aa 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.
- Cp.CH4.HW97.csv, V.CH4.HWM96.csv Apparent molar heat capacities and volumes of CH4 in dilute aqueous solutions reported by Hnědkovský and Wood, 1997 and Hnědkovský et al., 1996. See EOSregress and the vignette eos-regress.Rmd 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.Rmd 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.Rmd` 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 the vignette `anintro.Rmd` for an example that uses 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 `demos("AkDi")`.
- `TKSS14_Fig2.csv` Experimental data points digitized from Figure 2 of Tutolo et al., 2014, used in `demos("aluminum")`.
- `Mer75_Table4.csv` Values of log(aK+/aH+) and log(aNa+/aH+) from Table 4 of Merino, 1975, used in `demos("aluminum")`.

Files in `fasta` contain protein sequences:

- `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.Rmd` for an example that uses this file.

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

- `Sce.csv.xz` Data frame of amino acid composition of 6716 proteins from the *Saccharomyces* Genome Database (SGD). Values in the first three columns are the `ORF` names of proteins, `SGDID`, and `GENE` names. The remaining twenty columns (`ALA`..`VAL`) contain the numbers of the respective amino acids in each protein. The sources of data for '`Sce.csv`' are the files '`protein_properties.tab`' and '`SGD_features.tab`' (for the gene names), downloaded from `http://www.yeastgenome.org` on 2013-08-24. See `yeast.aa` for an example.
- `DS11.csv`, `DS13.csv` These two files contain amino acid compositions of metagenomically encoded proteins, averaged together according to functional annotation (DS11) or taxonomic affiliation (DS13). The data are from Dick and Shock, 2011 and 2013. They are used in the vignette *Hot-spring proteins in CHNOSZ*.
- `microbial.aa.csv` Overall protein compositions of microbial species reported to be positively or negatively enriched in colorectal cancer. This file is taken from Dick, 2016. It is used by `demo("bugstab")`.

Files in `refseq` contain code and results of processing NCBI Reference Sequences (RefSeq) for microbial proteins, using RefSeq release 61 of 2013-09-09:

- `README.txt` Instructions for producing the data files.
- `gencat.sh` Bash script to extract microbial protein records from the RefSeq catalog.

- gi.taxid.txt Output from above. The complete file is too large to distribute with CHNOSZ, but a portion is included in extdata/bison to support processing example BLAST files for the Bison Pool metagenome (based on RefSeq 57, 2013-01-08).
- mkfaa.sh Combine the contents of .faa.gz files into a single FASTA file (to use e.g. for making a BLAST database).
- protein.refseq.R Calculate average amino acid composition of all proteins for each organism identified by a taxonomic ID.
- trim_refseq.R Keep only selected organism names (reduces number of taxa from 6758 to 779, helps to control package size).
- protein_refseq.csv.xz Output from above. See example in pinfo.
- taxid.names.R Generate a table of scientific names for the provided taxids. Requires the complete names.dmp and nodes.dmp from NCBI taxonomy files.
- taxid_names.csv.xz Output from above. NOTE: For backward compatibility with the example BLAST files for the Bison Pool metagenome, the packaged file merges records for taxids found in either RefSeq 57 or 61. NOTE 2: To save space for the package, the file has been trimmed to hold only those taxids listed in extdata/bison/gi.taxid.txt. Certain taxids in release 57 were not located in the current RefSeq catalog, probably related to the transition to the "WP" multispecies accessions (ftp://ftp.ncbi.nlm.nih.gov/refseq/release/announcements/WP-proteins-06.10.2013.pdf). See example for id.blast.

Files in supcrt contain scripts for reading and comparing SUPCRT files (including slop98.dat and newer slop files from GEOPIG (http://geopig.asu.edu)) with the database in CHNOSZ:

- read.supcrt.R defines the function read.supcrt that can be used to read SUPCRT files.
- compare.R uses read.supcrt to compare data in the SUPCRT file with that in thermo()$obigt.
- newnames.csv maps names generated by read.supcrt, based on names present in the source SUPCRT files, to names used in thermo()$obigt.

Files in taxonomy contain taxonomic data files:

- names.dmp and nodes.dmp are excerpts of the taxonomy files available on the NCBI ftp site (ftp://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.

- DLEN67.csv Standard Gibbs energies of formation, in kcal/mol, from Dayhoff et al., 1967, for nitrogen (N2) plus 17 compounds shown in Fig. 2 of Dayhoff et al., 1964, at 300, 500, 700 and 1000 K. See demo("wjd") and the vignette wjd.Rmd for examples that use this file.
- SK95.csv contains thermodynamic data for alanate, glycinate, and their complexes with metals, taken from Shock and Koretsky (1995) as corrected in slop98.dat. The data are used in the package tests (test-recalculate.R) to check the recalculated values of G, H, and S in thermo()$obigt using properties for alanate and glycinate from Amend and Helgeson (1997).

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. https://doi.org/10.1016/S0016-7037(02)01141-9

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. https://doi.org/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. https://doi.org/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. https://doi.org/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. https://doi.org/10.1007/s004100050233

Dayhoff, M. O. and Lippincott, E. R. and Eck, R. V. (1964) Thermodynamic Equilibria In Prebiological Atmospheres. *Science* **146**, 1461–1464. https://doi.org/10.1126/science.146.3650.1461

Dayhoff, M. O. and Lippincott, E. R., Eck, R. V. and Nagarajan (1967) Thermodynamic Equilibrium In Prebiological Atmospheres of C, H, O, N, P, S, and Cl. Report SP-3040, National Aeronautics and Space Administration.

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

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

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. https://doi.org/10.1371/journal.pone.0022782

Dick, J. M. and Shock, E. L. (2013) A metastable equilibrium model for the relative abundance of microbial phyla in a hot spring. *PLoS ONE* **8**, e72395. https://doi.org/10.1371/journal.pone.0072395

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. https://doi.org/10.1016/S1476-9271(02)00094-4

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. https://doi.org/10.1038/nature01977

Ghaemmaghami, S., Huh, W., Bower, K., Howson, R. W., Belle, A., Dephoure, N., O'Shea, E. K. and Weissman, J. S. (2003) Global analysis of protein expression in yeast. *Nature* **425**(6959), 737–741. https://doi.org/10.1038/nature02046

Huh, W. K., Falvo, J. V., Gerke, L. C., Carroll, A. S., Howson, R. W., Weissman, J. S. and O'Shea, E. K. (2003) Global analysis of protein localization in budding yeast. *Nature* **425**(6959), 686–691. https://doi.org/10.1038/nature02026

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. https://doi.org/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. https://doi.org/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/)

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. https://doi.org/10.1016/0016-7037(75)90085-X

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. https://doi.org/10.1016/S0022-2836(05)80198-6

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. https://doi.org/10.1016/S0016-7037(97)00345-1

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. http://www.worldcat.org/oclc/32590140

Roxby, R. and Tanford, C. (1971) Hydrogen ion titration curve of lysozyme in 6 M guanidine hydrochloride. *Biochemistry* **10**, 3348–3352. https://doi.org/10.1021/bi00794a005

SGD project. *Saccharomyces* Genome Database, http://www.yeastgenome.org

Shock, E. and Canovas, P. (2010) The potential for abiotic organic synthesis and biosynthesis at seafloor hydrothermal systems. *Geofluids* **10**, 161–192. https://doi.org/10.1111/j.1468-8123.2010.00277.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. https://doi.org/10.1016/0016-7037(95)00058-8

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. https://doi.org/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. https://doi.org/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. https://doi.org/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. https://doi.org/10.1016/j.gca.2014.02.036

YeastGFP project. Yeast GFP Fusion Localization Database, http://yeastgfp.ucsf.edu; Current location: http://yeastgfp.yeastgenome.org

---

`findit` *Gridded Search to Optimize Objective Functions*

Description

Use a gridded search to find a combination of one or more of chemical activities of basis species, temperature and/or pressure that maximize or minimize a objective function of the metastable equilibrium chemical activities of the species of interest.

Usage

```
findit (lims = list(), objective = "CV", niter = NULL, iprotein = NULL,
  plot.it = TRUE, T = 25, P = "Psat", res = NULL, labcex = 0.6,
  loga2 = NULL, loga.balance = 0, rat = NULL,
  balance = NULL, normalize = FALSE)
plot_findit(x, which=NULL, mar=c(3.5,5,2,2), xlab="iteration", ...)
```

Arguments

`lims`	list, specification of search limits
`objective`	character, name of objective function to optimize
`niter`	numeric, number of iterations
`res`	numeric, grid resolution (number of points on one edge)
`iprotein`	numeric, indices of proteins
`plot.it`	logical, make a plot?
`T`	numeric, temperature

P	numeric, pressure; or character, "Psat"
labcex	numeric, character expansion for plot labels
loga2	numeric, reference logarithms of activity of species
loga.balance	numeric, logarithm of total activity of balanced quantity (passed to diagram)
rat	numeric, ratio of edge length in successive iterations
balance	character or numeric, balanced quantity (passed to diagram)
normalize	logical, normalize chemical formulas by the balance vector? (passed to diagram)
x	list, object of class findit
which	numeric, which of the parameters to plot
mar	numeric, plot margin specification
xlab	character, x-axis label
...	additional arguments passed to plot

Details

findit implements a gridded optimization to find the minimum or maximum value of an objective function. The variables are one or more of the chemical activities, temperature and/or pressure whose ranges are listed in lims. Generally, the system (basis species and species of interest) must be set up before calling this function. If iprotein is supplied, indicating a set of proteins to use in the calculation, the definition of the species is not required. lims is a list, each element of which is vector having a name that is the formula of one of the basis species, 'T' or 'P' and a pair of values indicating the range of the named parameter. The values are the logarithms of activities of the basis species, or temperature or pressure (in the user's units; see util.units). If either 'T' or 'P' is missing from the list in lims, the calculations are performed at isothermal and/or isobaric conditions indicated by T and P arguments.

Taking nd as the number of dimensions (number of variables in lims), default values of niter and res come from the following table. These settings have been selected to be able to run the function quickly in the higher dimensions. Detailed studies of a system might have to use more iterations and/or higher resolutions.

nd	niter	res	grid points (res^nd)	rat
1	4	128	128	0.7
2	6	64	4096	0.7
3	6	16	4096	0.8
4	8	8	4096	0.9
5	12	6	7776	0.9
6	12	4	4096	0.95
7	12	4	16384	0.95

The function performs niter iterations. At first, the limits of the parameters given in lims define the extent of a nd-dimensional box around the space of interest. The value of objective is calculated at each of the res^{nd} grid points and and optimum value located (see revisit). In the next iteration the new search box is centered on the location of the optimum value, and the edges are shrunk so their length is rat * the length in the previous step. If the limits of any of the parameters extend

beyond those in lims, they are pushed in to fit (preserving the difference between them).

plot_findit plots the values of the parameters and the objective function as a function of the number of iterations.

Value

findit returns a list having class findit with elements value (values of the parameters, and value of the objective function, at each iteration), lolim (lower limits of the parameters) and hilim (upper limits of the parameters).

See Also

demo("findit") and test-findit.R for examples.

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. https://doi.org/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, 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 searches of the indicated species are made among 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 thermo()$obigt is grouped by states in the order 'aq', 'cr', 'gas', 'liq', so for species in both aqueous and gaseous states the index of the aqueous species is returned, unless state is set to '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 protein, 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()$obigt indicated by ispecies are returned, after removing any order-of-magnitude scaling factors. If these species are all aqueous or are all not aqueous, the compounded column names used in thermo()$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 checkGHS and checkEOS.

See Also

thermo, check.obigt

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 record
info(si)
# it is possible to get a range of records
info(si:(si+3))

## dealing with states
```

```
# default order of preference for names: aq > gas > cr,liq
info(c("methane","ethanol","glycinate"))  # aq, aq, aq
info(c("adenosine","alanine","hydroxyapatite"))  # aq, aq, cr
# state argument overrides the default
info(c("ethanol","adenosine"),state=c("gas","cr"))
# formulas default to aqueous species, if available
info(c("CH4","CO2","CS2","MgO"))  # aq, aq, gas, cr
# state argument overrides the default
info(c("CH4","CO2","MgO"),"gas")  # gas, gas, NA
# exceptions to the aqueous default is O2
info("O2")  # gas

## 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. http://www.biogeosciences.net/3/311/2006/bg-3-311-2006.html

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. https://doi.org/10.1016/S0022-2836(05)80197-4

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. https://doi.org/10.1016/S0022-2836(05)80198-6

See Also

`pinfo`, `affinity`

Examples

```
## Charge of LYSC_CHICK as a function of pH and T
# After Fig. 10 of Dick et al., 2006
# the rownumber of the protein in thermo()$protein
ip <- pinfo("LYSC_CHICK")
# its amino acid composition
aa <- pinfo(ip)
# additive charges of unfolded protein at 25, 100, 150 degrees C
# as a function of pH
```

```
pH <- seq(0, 14, 0.1)
Z.25 <- ionize.aa(aa, T=25, pH=pH)
plot(pH, Z.25[, 1], type="l", xlab="pH", ylab="net charge (Z)")
lines(pH, ionize.aa(aa, T=100, pH=pH)[, 1], col="red")
lines(pH, ionize.aa(aa, T=150, pH=pH)[, 1], col="orange")
text(c(12, 10, 9), c(-15, -16, -18),
  labels=paste("T=", c(25, 100, 150), sep=""))
# suppress ionization of cysteine as if it was oxidized
# (to form non-ionizable cystine disulfide bonds)
lines(pH, ionize.aa(aa, T=25, pH=pH, suppress.Cys=TRUE)[, 1], lty=2)
text(12, -7, "T=25, oxidized")
# add experimental points
RT71 <- read.csv(system.file("extdata/cpetc/RT71.csv", package="CHNOSZ"))
points(RT71$pH, RT71$Z)
legend("topright", pch=1, legend="Roxby and Tanford, 1971")
title(main=paste("Ionization of unfolded LYSC_CHICK\n",
  "After Dick et al., 2006"))

## 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(5000, 8000))
# 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, convert(MP90.cp("LYSC_CHICK", T), "cal"))
# Privalov and Makhatadze's experimental values
e <- read.csv(system.file("extdata/cpetc/PM90.csv", package="CHNOSZ"))
points(e$T, convert(e$LYSC_CHICK, "cal"), 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")
```

---

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

```
# the composition of a simple compound
makeup("CO2")     # 1 carbon, 2 oxygen
# the formula of lawsonite, with a parenthetical part and a suffix
makeup("CaAl2Si2O7(OH)2*H2O")
# fractional coefficients are ok
redfield <- c(106, 16, 1)
reddiv10 <- makeup("C10.6N1.6P0.1")
stopifnot(10*reddiv10 == redfield)

# 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))
stopifnot(all(mf==0))
```

`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, bases2 = NULL, blend = TRUE, mixing = TRUE, ...)
```

Arguments

bases	character, basis species to be changed in the calculation, or list, containing vectors for each group of changing basis species
bases2	character, second set of changing basis species
blend	logical, use relative abundances of basis species?
mixing	logical, include a term for the Gibbs energy of mixing?
...	additional arguments to be passed to affinity

Details

mosaic can be used to calculate the affinities of formation of species when the relative abundances of basis species listed in bases changes 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 basis species can described as "mosaic diagrams".

The function calculates the affinities using all combination of basis species given as vector arguments to bases and bases2. 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). A second, independent set of basis species can be provided in bases2 (for example 'CO3-2', 'HCO3-', 'CO2', if the first set of basis species are the sulfur-bearing ones listed above). The arguments in ... are passed to affinity to specify the conditions.

If blend is TRUE (the default), the function combines the affinities of the formation reactions in proportion to the relative abundances of the basis species at each condition. Additionally, if mixing is TRUE (the default), a term is included to account for the Gibbs energy of ideal mixing. See the second example in solubility for a numerical test of the calculations using blend and mixing. If blend is FALSE, the function returns the affinities calculated using the single predominant basis species in bases at each condition (in this case, the mixing argument has no effect).

A more flexible method of specifying multiple sets of basis species is now available. Instead of using bases and bases2, supply a list for just the bases argument. The list should contain any number of vectors specifying the groups of basis species. All combinations of basis species in these groups are used for the calculations. This overcomes the prior limitation of only having two changing groups of basis species.

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 bases2 is provided, the list also contains A.bases2 (affinities of formation of the second set of basis species).

References

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

See Also

demo("mosaic"), extending the example below by addition of carbonate species in bases2, and using thermodynamic data from Garrels and Christ, 1965.

Examples

```
# Fe-minerals and aqueous species in Fe-S-O-H system
# speciate SO4-2, HSO4-, HS-, H2S as a function of Eh and pH
# after Garrels and Christ, 1965 Figure 7.20
pH <- c(0, 14, 250)
Eh <- c(-1, 1, 250)
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"))
# 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 = NULL)
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 basis species predominance fields
#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(T = seq(100, 500, 100), P = 1000, m_tot = 2, ...)
```

Arguments

T	numeric, temperature in °C
P	numeric, pressure in bar (single value)
m_tot	numeric, total molality of NaCl (single value)
...	additional arguments for subcrt

Details

This function calculates speciation (ion activities) and ionic strength in aqueous solutions given a total amount (m_tot, in mol/kg) of NaCl. The function is written for quick calculations along a temperature range (T) at constant pressure (P). The only reaction considered is $Na^+ + Cl^- =$ NaCl(aq). The algorithm starts by calculating the equilibrium constant (K) of the reaction and assuming complete dissociation of NaCl(aq). This also permits calculating the ionic strength from the molalities of Na^+ and Cl^-. Then, uniroot is used to find the equilibrium molality of Cl^-; that is, where the affinity of the reaction ($\log(K/Q)$) becomes zero. The activity quotient (Q) is evaluated taking account of activity coefficients of Na^+, Cl^-, and NaCl(aq) calculated for the nominal ionic strength (see nonideal). The calculated molality of Cl^- yields a new estimate of the ionic strength of the system. The calculations are iterated until the deviation in ionic strength at all temperatures is less than 0.01.

Value

A list with components 'IS' ("true" ionic strength from concentrations of unpaired ions), 'm_Cl' (molality of Cl^-), 'gam_Na', and 'gam_Cl' (activity coefficients of Na^+ and Cl^-).

Warning

This function provides only a first-order estimate of the solution composition, and is intended for solubility calculations of relatively insoluble metals in NaCl-dominated solutions. The formation of other species such as HCl or NaOH is not accounted for.

References

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

See Also

demo("gold") for an application of this function.

Examples

```
# ionic strength of solution and activity coefficient of Cl-
# from 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))
gam_Cl.HCh <- list(`100`=c(0.565, 0.545, 0.551, 0.567, 0.589, 0.615),
                   `300`=c(0.366, 0.307, 0.275, 0.254, 0.238, 0.224),
                   `500`=c(0.19, 0.137, 0.111, 0.096, 0.085, 0.077))
# 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))
gam_Cl.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(c(100, 300, 500), P=1000, m_tot=m_tot[i])
  IS.calc[i, ] <- NaCl.out$IS
  gam_Cl.calc[i, ] <- NaCl.out$gam_Cl
}
# plot ionic strength from HCh and NaCl() as points and lines
opar <- par(mfrow=c(2, 1))
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 1:1 line, legend, and title
abline(0, 1, lty=3)
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)")
plot(c(1,6), c(0,0.8), xlab="NaCl (mol/kg)", ylab=expression(gamma[Cl^"-"]), type="n")
# plot activity coefficient (gamma)
for(i in 1:3) {
  points(m.HCh, gam_Cl.HCh[[i]], col=col[i])
  lines(m_tot, gam_Cl.calc[, i], col=col[i])
}
# we should be fairly close
stopifnot(maxdiff(unlist(gam_Cl.calc[seq(1,11,2), ]), unlist(gam_Cl.HCh)) < 0.033)
par(opar)
```

`nonideal` *Activity Coefficients of Aqueous Species*

Description

Calculate activity coefficients and adjusted (transformed) 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	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, 'Alberty', 'Bdot', 'Bdot0', or 'bgamma'
TC	numeric, temperature (°C)
showsplines	character, show isobaric ('T') or isothermal ('P') splines

Details

This function calculates activity coefficients and adjusted thermodynamic proeprties (i.e. transformed standard Gibbs energy) for charged and neutral aqueous species. The species are 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 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.

Value

One ('G') or more ('H', 'S', 'Cp'; currently only with the Alberty method) standard thermodynamic properties (at IS=0) in speciesprops are replaced by the corresponding adjusted thermodynamic 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.

Charged Species

Calculations for the proton (H^+) and electron (e^-) are skipped by default; 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 thermooptideal.H or ideal.e to FALSE.

If method is 'Alberty', the values of IS are combined with Alberty's (2003) equation 3.6-1 (extended Debye-Hückel equation) and its derivatives, to calculate adjusted molal properties at the specified ionic strength(s) and temperature(s). The adjusted molal properties that can be calculated include 'G', 'H', 'S' and 'Cp'; any columns in the dataframes of speciesprops with other names are left untouched.

In addition to IS and T, the following two 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).

If method is 'Bdot' (the default setting in CHNOSZ), the "B-dot" form of the extended Debye-Hückel equation equation is used. The distance of closest approach (the "ion size parameter") is taken from the UT_SIZES.REF file of the HCh package (Shvarov and Bastrakov, 1992), which is based on Table 2.7 of Garrels and Christ, 1965. The extended term parameter is expressed as "B-dot", which is a function only of temperature (Helgeson, 1969). For some uses, it is desirable to set the extended term parameter to zero; this can be done by setting the method to 'Bdot0'.

If method is 'bgamma', 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 thermooptSetchenow 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()optSetchenow is 'bgamma', the extended term paramter is taken from the setting for the charged species (which can be either 'Bdot' or 'bgamma'). Set thermo()optSetchenow to any other value to disable the calculations for neutral species.

b_gamma

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 crude method of kriging the data, but produces fairly smooth interpolations without adding any external dependencies.

References

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

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

Helgeson, H. C. (1969) Thermodynamics of hydrothermal systems at elevated temperatures and pressures. *Am. J. Sci.* **267**, 729–804. https://doi.org/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. https://doi.org/10.2475/ajs.281.10.1249

Manning, C. E. (2013) Thermodynamic modeling of fluid-rock interaction at mid-crustal to upper-mantle conditions. *Rev. Mineral. Geochem.* **76**, 135–164. https://doi.org/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. https://doi.org/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**. http://pid.geoscience.gov.au/dataset/ga/25473

Examples

```
### Examples following Alberty, 2003
### (page numbers given below)

## the default method setting is Bdot;
## change it to Alberty
oldnon <- nonideal("Alberty")

## using nonideal() directly
# p. 273-276: activity coefficient (gamma)
# as a function of ionic strength and temperature
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)))
```

```
## more often, the 'IS' argument of subcrt() is used to compute
## adjusted properties at given ionic strength
# p. 16 Table 1.3: adjusted pKa of acetic acid
# set ideal.H to FALSE to calculate activity coefficients for the proton
# (makes for better replication of the values in Alberty's book)
thermo("opt$ideal.H" = FALSE)
(sres <- subcrt(c("acetate", "H+", "acetic acid"), c(-1, -1, 1),
  IS=c(0, 0.1, 0.25), T=25, property="logK"))
# we're within 0.01 log of Alberty's pK values
Alberty_logK <- c(4.75, 4.54, 4.47)
stopifnot(maxdiff(sres$out$logK, Alberty_logK) < 0.01)
# reset option to default
thermo("opt$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=NULL, 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=NULL, col="red")
par(opar)


### finished with Alberty equation, let's look at b_gamma
nonideal("bgamma")

## activity coefficients for monovalent ions at 700 degC, 10 kbar
# after Manning, 2013, Fig. 7
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")
wprop <- water(c("A_DH", "B_DH"), T=convert(700, "K"), P=10000)
water(oldwat)
# just use an empty table for a single species
speciesprops <- list(data.frame(G=rep(0, length(IS))))
# choose any monovalent species
(nonidealprops <- nonideal("Na+", speciesprops, IS=IS,
```

```
  T=convert(700, "K"), P=10000, A_DH=wprop$A_DH, B_DH=wprop$B_DH))
# we get the nonideal Gibbs energy contribution and
# the activity coefficient; check values of the latter
Manning_gamma <- c(0.93, 0.82, 0.65, 0.76, 1.28, 2)
gamma <- 10^nonidealprops[[1]]$loggam
# the error is larger at higher IS
stopifnot(maxdiff(gamma[1], Manning_gamma[1]) < 0.01)
stopifnot(maxdiff(gamma, Manning_gamma) < 0.23)

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

# done with examples, restore default setting
nonideal(oldnon) # == nonideal("Bdot")
```

`objective` *Objective Functions*

Description

Calculate statistical and thermodynamic quantities for activities of species. These functions can be specified as objectives in `revisit` and `findit` in order to identify optimal chemical conditions.

Usage

```
SD(a1)
CV(a1)
shannon(a1)
DGmix(loga1)
qqr(loga1)
logact(loga1, loga2)
spearman(loga1, loga2)
pearson(loga1, loga2)
RMSD(loga1, loga2)
CVRMSD(loga1, loga2)
DDGmix(loga1, loga2)
DGinf(a1, a2)
DGtr(loga1, loga2, Astar)
```

Arguments

`a1`	numeric matrix, chemical activities of species
`loga1`	numeric matrix, logarithms of activity
`loga2`	numeric, reference values of logarithms of activity
`a2`	numeric, reference values of activity
`Astar`	numeric, reference values of chemical affinity

Details

The value in a1 or loga1 is a matrix of chemical activities or logarithms of activity with a column for each species, and a row for each chemical condition. Except for calculations of the Shannon entropy, all logarithmic bases (including in the equations below) are decimal.

SD, CV and shannon calculate the standard deviation, coefficient of variation, and Shannon entropy for the values in each row of a1. The Shannon entropy is calculated from the fractional abundances: H = sum(-p * log(p)) (natural logarithm), where p=a1/sum(a1).

DGmix calculates the Gibbs energy/2.303RT of ideal mixing from pure components corresponding to one molal (unit activity) solutions: DGmix/2.303RT = sum(a1 * loga1) (cf. Eq. 7.20 of Anderson, 2005).

qqr calculates the correlation coefficient on a quantile-quantile (Q-Q) plot (see qqnorm) for each row of loga1, giving some indication of the resemblance of the chemical activities to a log-normal distribution.

logact returns the logarithm of activity of a single species identified by index in loga2 (which of the species in the system).

spearman, pearson, RMSD and CVRMSD calculate Spearman's rank correlation coefficient, the Pearson correlation coefficient, the root mean sqaured deviation (RMSD) and the coefficient of variation of the RMSD between each row of loga1 and the values in loga2. The CVRMSD is computed as the RMSD divided by the mean of the values in loga1.

DDGmix calculates the difference in Gibbs energy/2.303RT of ideal mixing between the assemblages with logarithms of activity loga1 and loga2.

DGinf calculates the difference in Gibbs energy/2.303RT attributed to relative informatic entropy between an initial assemblage with activities a2 and final assemblage(s) with activities with activities in each row of a1. The equation used is DGinf/2.303RT = sum(p2 * (logp2 - logp1)), where p1 and p2 are the proportions, i.e. p1 = a1 / sum(a1) and p2 = a2 / sum(a2). This equation has the form of the Kullback-Leibler divergence, sometimes known as relative entropy (Ludovisi and Taticchi, 2006). In specific cases (systems where formulas of species are normalized by the balancing coefficients), the values of DGinf and DGtr are equal.

DGtr calculates the change in Gibbs energy/2.303RT of a system in which species with initial logarithms of activitiy (loga1) are transformed to the same species with different final logarithms of activity (loga2) at constant temperature, pressure and chemical potentials of basis species. It is calculated as the sum over species of (G2-G1) where G1/RT = -a1*Astar + a1*loga1 - a1 + a constant (where a1 is 10^loga1), likewise for G2, and where Astar is the starved affinity, that is the affinity of the reaction to form one mole of the species at unit activity from the basis species in their defined activities. The equation used arises from integrating dG = -A/dxi = -A/dn where xi is the reaction progress variable, dn/dxi = 1 is the reaction coefficient on the species, and A = Astar - 2.303RTloga is the chemical affinity (Dick and Shock, 2013).

Each objective function has an attribute (see attributes and structure) named 'optimum' that takes the value of 'minimal' (SD, CV, RMSD, CVRMSD, DGmix, DDGmix, DGtr) or 'maximal' (logact, shannon, qqr, spearman, pearson).

References

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

Dick, J. M. and Shock, E. L. (2013) A metastable equilibrium model for the relative abundance of microbial phyla in a hot spring. *PLoS ONE* **8**, e72395. https://doi.org/10.1371/journal.pone.0072395

Ludovisi, A. and Taticchi, M. I. (2006) Investigating beta diversity by Kullback-Leibler information measures. *Ecological Modelling* **192**, 299–313. https://doi.org/10.1016/j.ecolmodel.2005.05.022

See Also

revisit, findit

Examples

```
## a made-up system: 4 species, 1 condition
loga1 <- t(-4:-1)
loga2 <- loga1 + 1
stopifnot(qqr(loga1) < 1)
stopifnot(RMSD(loga1, loga1) == 0)
stopifnot(RMSD(loga1, loga2) == 1)
stopifnot(CVRMSD(loga1, loga2) == -0.4) # 1 / mean(-4:-1)
stopifnot(spearman(loga1, loga2) == 1)
stopifnot(spearman(loga1, rev(loga2)) == -1)
# less statistical, more thermodynamical...
stopifnot(all.equal(DGmix(loga1), -0.1234)) # as expected for decimal logarithms
stopifnot(all.equal(DDGmix(loga1, loga2), 0.0004))

## transforming an equilibrium assemblage of n-alkanes
basis(c("CH4", "H2"), c("gas", "gas"))
species(c("methane", "ethane", "propane", "butane"), "liq")
# calculate equilibrium assemblages over a range of logaH2
a <- affinity(H2=c(-10, -5, 101), exceed.Ttr=TRUE)
e <- equilibrate(a)
# take a reference equilibrium distribution at logfH2 = -7.5
loga1 <- list2array(e$loga.equil)[51, ]
Astar <- list2array(e$Astar)[51, ]
# equilibrium at any other logfH2 is not equilibrium at logfH2 = -7.5
DGtr.out <- DDGmix.out <- numeric()
for(i in 1:length(a$vals[[1]])) {
  loga2 <- list2array(e$loga.equil)[i, ]
  DGtr.out <- c(DGtr.out, DGtr(t(loga1), loga2, t(Astar)))
  DDGmix.out <- c(DDGmix.out, DDGmix(t(loga1), loga2))
}
# all(DGtr >= 0) is TRUE
stopifnot(all(DGtr.out >= 0))
# all(DDGmix >= 0) is FALSE
stopifnot(!all(DDGmix.out >= 0))
# a plot is also nice
thermo.plot.new(xlim=range(a$vals[[1]]), xlab=axis.label("H2"),
  ylim=range(DDGmix.out, DGtr.out), ylab="energy")
abline(h=0, lty=2)
abline(v=-7.5, lty=2)
```

```
text(-7.6, 2, "reference condition", srt=90)
lines(a$vals[[1]], DDGmix.out)
lines(a$vals[[1]], DGtr.out)
text(-6, 5.5, expr.property("DDGmix/2.303RT"))
text(-6, 2.3, expr.property("DGtr/2.303RT"))
title(main=paste("Transformation between metastable equilibrium\n",
  "assemblages of n-alkanes"))
# take-home message: use DGtr to measure distance from equilibrium in
# open-system transformations (constant T, P, chemical potentials of basis species)
```

---

palply *Conditional Parallel Processing*

Description

Use multiple processors for large calculations.

Usage

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

Arguments

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

Details

palply is a wrapper function to run parLapply if length of X > thermooptparamin 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 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 'demos/density.R'.

protein *Examples of Calculations for Proteins*

Description

This page contains some examples of using the functions in CHNOSZ to calculate thermodynamic properties of and make diagrams for proteins.

Examples

```
## Standard molal entropy of a protein reaction
basis("CHNOS")
# here we provide the reaction coefficients of the
# proteins (per protein backbone); subcrt() calculates
# the coefficients of the basis species in the reaction
s <- subcrt(c("CSG_METTL", "CSG_METJA"), c(-1/530, 1/530),
  T=seq(0, 350, length.out=50))
# note: this uses the properties of the nonionized proteins

## logfO2-pH potential diagram
# with a charged basis, we calculate properties of ionized proteins
basis("CHNOS+")
file <- system.file("extdata/protein/DS11.csv", package = "CHNOSZ")
aa <- read.csv(file, as.is=TRUE)
aa <- aa[grep("transferase", aa$protein), ]
ip <- add.protein(aa)
a <- affinity(pH=c(0, 14), O2=c(-64, -61), T=75, iprotein=ip)
diagram(a)
title(main="Sequences for transferase at Bison Pool")

## surface-layer proteins from Methanococcus and others
## as a function of oxygen fugacity, after Dick, 2008, Fig. 5b
# to reproduce the calculations in the paper,
# use superseded data for [Met], [Gly] and [UPBB]
reset()
add.obigt("OldAA")
# make our protein list
organisms <- c("METSC", "METJA", "METFE", "HALJP", "METVO",
  "METBU", "ACEKI", "GEOSE", "BACLI", "AERSA")
proteins <- c(rep("CSG", 6), rep("SLAP", 4))
proteins <- paste(proteins, organisms, sep="_")
# load the basis species and proteins
basis("CHNOS+")
species(proteins)
# calculate affinities; we go to lower logfO2 than Dick, 2008
# and find an interesting convergence of stabilities there
a <- affinity(O2=c(-100, -65))
# try normalize=FALSE to make Fig. 5a in the paper
e <- equilibrate(a, normalize=TRUE)
```

```
d <- diagram(e, ylim=c(-5, -1), names=organisms, format.names=FALSE)
# add water stability line
abline(v=-83.1, lty=2)
title(main="Surface-layer proteins, after Dick, 2008")
# checking the geometry of the diagram
# most preominant along the x-axis
stopifnot(organisms[unique(which.pmax(e$loga.equil))] ==
  c("METFE", "METJA", "METVO", "HALJP"))
# stability order close to logfO2=-83.1
stopifnot(order(as.data.frame(e$loga.equil)[62,],
  decreasing=TRUE)==c(2, 6, 7, 5, 3, 1, 9, 8, 10, 4))
# reset thermodynamic database
reset()

## relative stabilities of bovine proteins
## as a function of temperature along a glutathione redox buffer
mod.buffer("GSH-GSSG", c("GSH","GSSG"), logact=c(-3, -7))
basis(c("CO2", "H2O", "NH4+", "SO4-2", "H2", "H+"),
  c(-1, 0, -4, -4, 999, -7))
basis("H2", "GSH-GSSG")
basis("CO2", "gas")
prot <- c("CYC", "RNAS1", "BPT1", "ALBU", "INS", "PRIO")
species(prot, "BOVIN")
a <- affinity(T=c(0, 200))
# set line colors according to oxidation state of carbon
ZC <- ZC(species()$ispecies)
col <- ZC.col(ZC)
e <- equilibrate(a, normalize=TRUE)
d <- diagram(e, col=col, lwd=3)
title(main="Bovine proteins, GSH/GSSG redox buffer")
```

`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. https://doi.org/10.5194/bg-3-311-2006

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

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

Examples

```
# search by name in thermo()$protein
ip1 <- pinfo("LYSC_CHICK")
ip2 <- pinfo("LYSC", "CHICK")
# these are the same
stopifnot(all.equal(ip1, ip2))
# 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 example of an unrecognized protein name
ip4 <- pinfo("MYGPHYCA")
stopifnot(is.na(ip4))

## example for chicken lysozyme C
# index in thermo()$protein
ip <- pinfo("LYSC_CHICK")
```

```
# amino acid composition
pinfo(ip)
# 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")
# affinity calculation, protein identified by ip
basis("CHNOS+")
affinity(iprotein=ip)
# affinity calculation, protein loaded as a species
species("LYSC_CHICK")
affinity()
# NB: 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")))

## using protein.formula: average oxidation state of
## carbon of proteins from different organisms (Dick, 2014)
# get amino acid compositions of microbial proteins
# generated from the RefSeq database
file <- system.file("extdata/refseq/protein_refseq.csv.xz", package="CHNOSZ")
ip <- add.protein(read.csv(file, as.is=TRUE))
# only use those organisms with a certain
# number of sequenced bases
ip <- ip[as.numeric(thermo()$protein$abbrv[ip]) > 50000]
pf <- protein.formula(thermo()$protein[ip, ])
zc <- ZC(pf)
# the organism names we search for
# "" matches all organisms
terms <- c("Natr", "Halo", "Rhodo", "Acido", "Methylo",
  "Chloro", "Nitro", "Desulfo", "Geo", "Methano",
  "Thermo", "Pyro", "Sulfo", "Buchner", "")
tps <- thermo()$protein$ref[ip]
plot(0, 0, xlim=c(1, 15), ylim=c(-0.3, -0.05), pch="",
  ylab=expression(italic(Z)[C]),
  xlab="", xaxt="n", mar=c(6, 3, 1, 1))
for(i in 1:length(terms)) {
  it <- grep(terms[i], tps)
  zct <- zc[it]
  points(jitter(rep(i, length(zct))), zct, pch=20)
}
terms[15] <- paste("all", length(ip))
axis(1, 1:15, terms, las=2)
```

```
title(main=paste("Average oxidation state of carbon in proteins",
  "by taxID in NCBI RefSeq (after Dick, 2014)", sep="\n"))


# using pinfo() with regexp=TRUE:
# plot ZC and nH2O/residue of HOX proteins
# basis species: glutamine-glutamic acid-cysteine-O2-H2O
basis("QEC")
# device setup
par(mfrow=c(2, 2))
# a red-blue scale from 1-13
col <- ZC.col(1:13)
# axis labels
ZClab <- axis.label("ZC")
nH2Olab <- expression(bar(italic(n))[H[2]*O])
# loop over HOX gene clusters
for(cluster in c("A", "B", "C", "D")) {
  # get protein indices
  pattern <- paste0("^HX", cluster)
  ip <- pinfo(pattern, "HUMAN", regexp=TRUE)
  # calculate ZC and nH2O/residue
  thisZC <- ZC(protein.formula(ip))
  thisH2O <- protein.basis(ip)[, "H2O"] / protein.length(ip)
  # plot lines
  plot(thisZC, thisH2O, type="l", xlab=ZClab, ylab=nH2Olab)
  # the number of the HOX gene
  pname <- pinfo(ip)$protein
  nHOX <- as.numeric(gsub("[A-Za-z]*", "", pname))
  # plot colored points
  points(thisZC, thisH2O, pch=19, col=col[nHOX], cex=3.5)
  points(thisZC, thisH2O, pch=19, col="white", cex=2.5)
  # plot the number of the HOX gene
  text(thisZC, thisH2O, nHOX)
  # add title
  title(main=paste0("HOX", cluster))
}
```

`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,
  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).
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.

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

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")

# an Eh-pH diagram for Mn-bearing aqueous species
basis(c("Mn+2", "H2O", "H+", "e-"))
iMn <- retrieve("Mn", c("O", "H"), "aq")
species(iMn)
a <- affinity(pH = c(6, 14), Eh = c(-1, 1))
diagram(a, fill = "terrain")
```

`revisit` *Plots and Optima of Objective Functions*

Description

Calculate values of an objective function from logarithms of activities of chemical species and (for some objectives) reference logarithms of activity. Make line or contour plots showing the values of the objective function and the positions of the optima (minimum or maximum).

Usage

```
revisit(eout, objective = "CV", loga2 = NULL, loga0 = NULL,
  ispecies = NULL, col = par("fg"), yline = 2, ylim = NULL,
  cex = par("cex"), lwd = par("lwd"), mar = NULL, side = 1:4,
  xlim = NULL, labcex = 0.6, pch = 1, main = NULL, plot.it = NULL,
  add = FALSE, plot.optval = TRUE, style.2D = "contour", bg = par("bg"))
```

Arguments

`eout`	list, output from `equilibrate`, containing logarithms of activities of species
`objective`	character, name of `objective` function
`loga2`	numeric vector, reference values of logarithm of activities
`loga0`	numeric vector, logarithm of activities to calculate activity ratios
`ispecies`	numeric, which species to consider
`col`	character, color to use for points or lines
`yline`	numeric, margin line for y-axis label
`ylim`	numeric, limits of y axis
`cex`	numeric, character expansion factor
`lwd`	numeric, line width

mar	numeric, plot margin specifications
side	numeric, which sides of plot to draw axes
xlim	numeric, limits of x axis
labcex	numeric, character expansion factor for species labels
pch	numeric, plotting symbol(s) to use for points
main	character, main title for plot
plot.it	logical, make a plot?
add	logical, add to an existing plot?
plot.optval	logical, show the location of the optimal value(s)?
style.2D	character, type of 2-D plot
bg	character, background for points

Details

revisit is used to calculate the variation in the equilibrium logarithms of chemical activity (supplied in eout) or to compare the calculated values with reference (e.g. measured) values (loga2). Usually, the output of equilibrate is used as the value for eout. The type of calculation is indicated by objective, giving the name of an objective function. Generally, loga2 is expressed in base-10 logarithms. However, if loga0 (base 10) is supplied, it is used to calculate the base-2 log ratio (log2(a1/a0)); these calculated values are then compared with values in loga2 interpreted as base-2 logarithms.

Internally, the list of logarithms of chemical activities in eout$loga.equil is passed as loga1 to the objective function. If the objective function has an argument a1 instead of loga1, the activities instead of their logarithms are passed to the function. Generally, loga2 must be a numeric vector with length equal to that of loga1 (i.e., number of species). However, if a single numeric value is supplied for loga2, it is recycled to the length of loga1.

For calculations at a single condition (0-D, no variation), with the 'qqr' objective, a quantile-quantile plot (qqnorm) is shown. For 'rmsd' and other objective functions having reference values (loga2), a scatter plot is shown with a smooth line calculated using loess.smooth. The line can be suppressed using lwd=NULL. Otherwise, no plot is made for 0-D calculations for the other objective functions.

If plot.it is TRUE, and eout is the output from equilibrate, and the number of variables is 1 or 2, the results are plotted — a line diagram in 1 dimension or a contour plot in 2 dimensions. style.2D can be set to image to fill the plot with colors instead of the contour plot that is the default.

If plot.optval is TRUE, the location of the optimum (or optima) is indicated by a dashed vertical line(s) on a 1-D plot or a point(s) marked by an asterisk on a 2-D plot. Also, on 2-D plots, the locations of the optima at each grid line perpendicular to the x and y axes are plotted. These points follow major ridges or valleys, and are plotted as dashed lines colored green for the x and blue for the y values.

An alternative source for the eout argument is any list of numeric values, each element of which corresponds to a different observation (such as a single species), all having the same dimensions (as vectors, matrices or higher-dimensional arrays) In this case, plotting is disabled, since the names of the variables are not in the input.

'revisit' is a partial anagram of 'diversity', which was the provisional name of the function but was changed in CHNOSZ-0.9. While the diversity function (in **vegan**) operates on a matrix with (biological) species on the columns, revisit operates on a list with (chemical) species as the elements of the list. The name of the 'H' output value is the conventional symbol for the Shannon diversity index, which was the first target statistic to be implemented in revisit.

Value

revisit returns a list containing at least an element named 'H', giving the calculated values of the objective function. For 1 or 2 dimensions of variability of chemical conditions, the output also contains the elements ixopt and iyopt (1-D and 2-D) and iyopt and yopt (2-D) indicating the positions and values of the optimum. The 'optimum' attribute of the objective function indicates whether minimal or maximal values are used. For calculations in more than two dimensions, the output contains iopt, a matrix.

See Also

demo("revisit") shows calculations for a system of proteins. findit is a related function implementing a gridded search of chemical activities, temperature and/or pressure that optimize the objective function.

Examples

```
## example of defining a new objective function
# count the species with logarithms of activity greater than loga2
count <- function(loga1, loga2) rowSums(loga1 > loga2)
# set the attribute indicating the type of optimum
attr(count, "optimum") <- "maximal"
# equilibrate a system of amino acids
basis("CHNOS")
species(aminoacids(""))
a <- affinity(O2=c(-80, -60))
e <- equilibrate(a)
# make a plot
r <- revisit(e, "count", -5)
title(main="Amino acids with metastable log activities > -5")

# can also make a 2-D plot
a <- affinity(O2=c(-74, -60, 25), H2O=c(-3, 3, 25))
e <- equilibrate(a)
r <- revisit(e, "count", -5, style.2D="image", plot.optval=FALSE)
title(main="Amino acids with metastable log activities > -5")

## 'revisit' calculations for amino acids
opar <- par(mfrow=c(2, 2))
basis("CHNOS+")
species(aminoacids(""))
# chemical affinities as a function of logarithm of oxygen fugacity
a <- affinity(O2=c(-85, -60))
# shows the equilibrium abundances of the amino acids
e <- equilibrate(a)
```

```
diagram(e)
mtitle(c("20 amino acids", "balanced on CO2"))
# show a legend with input constraints
db <- describe.basis(ibasis=3)
dp <- describe.property("T", 25)
legend("bottomright", c(dp, db))
# default is to plot coefficient of variation
r <- revisit(e)
# show a title with the optimal conditions
mincv <- format(r$optimum, digits=3)
t1 <- paste("minimum coeff of variation,", mincv, "at:")
# the logfO2 that minimized the C.V.
basis("O2", r$x)
t2 <- describe.basis(ibasis=5)
mtitle(c(t1, as.expression(t2)))
# chemical affinities as a function of two other variables
a <- affinity(NH3=c(-10, 10, 40), T=c(0, 80, 40))
diagram(a, fill="heat")
# show a legend with input constraints
db <- describe.basis(ibasis=5)
legend("bottomright", as.expression(db))
# contour plot of the CV
e <- equilibrate(a)
r <- revisit(e)
# show a title with the optimal conditions
mincv <- format(r$optimum, digits=3)
t1 <- paste("minimum coeff of variation,", mincv, "at:")
# the logaNH3 and T that minimized the C.V.
basis("NH3", r$x)
db <- describe.basis(ibasis=3)
dp <- describe.property("T", r$y)
t2 <- substitute(list(dp, db), list(dp=dp[[1]], db=db[[1]]))
mtitle(c(t1, as.expression(t2)))
par(opar)
```

`solubility` *Equilibrium Chemical Activities of Species*

Description

Calculate chemical activities of species in equilibrium with a soluble basis species.

Usage

```
solubility(aout, dissociation = NULL, find.IS = FALSE, in.terms.of = NULL)
```

Arguments

`aout`	list, output from `affinity`
`dissociation`	logical, does the mineral undergo a dissociation reaction?

find.IS	logical, find the equilibrium ionic strength by iteration?
in.terms.of	character, express the total solubility in terms of moles of this species

Details

This function performs a simple task: from the values of affinity of formation reactions of species at given activity, it works backward to find the activities of species that make the affinities zero. This corresponds to complete equilibrium with all of the 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). Internally, this is treated as the conserved basis species, so it must be present in all of the formation reactions of the species. It is also possible to set the conserved basis species as other than the first one (see demo(gold)), but this implies that dissociation reactions are not occurring (see below).

The species should be defined to represent one set of ions (anions or cations or their 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).

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 accomplished 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}.

The function attempts to automatically detect whether dissociation reactions are involved by looking at the first species. If the formation reaction of that species includes *both* the first and second basis species, the dissociation flag is set to TRUE. An example reaction of this type can be found in demo(solubility): CaCO3 (first basis species) = Ca+2 (second basis species) + CO3-2 (first species). Note that if the conserved basis species is not the first basis species, then the automatic detection of dissociation will always return FALSE. Therefore, a reaction corresponding to Au (fourth basis species) + ... = ... gives dissociation = FALSE (see demo(gold)). This algorithm for determining whether dissociation occurs is prone to error, so dissociation can be explicitly set in the arguments. A *not recommended* alternative is to set dissociation to a numeric value corresponding to the stoichiometry of released 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.

Note that other variables (pH, ionic strength, activities of other basis species) should be defined in the preceding call to affinity. However, for dissolving a substance in pure water, find.IS can be set to TRUE to determine the final ionic strength. This works by calculating the ionic strength from the equilibrium solubility calculation, then re-running affinity with those values. 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 accomplished by using the mosaic function instead of affinity. See the second example for this method.

The output of solubility has the same format as that of equilibrate, and can be used by diagram with type = "loga.balance" to plot the solubilities, or with type = NULL to plot the activities

of species. The value of `loga.balance` reflects the activity (or molality) of the conserved basis species, i.e. the thing being dissolved. Use `in.terms.of` to express this value in terms of another species. For example, the solubility of corundum (Al_2O_3) can be expressed in terms of the moles of Al^{+3} in solution (see the vignette `anintro.Rmd`).

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.

References

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. https://doi.org/10.2138/rmg.2013.75.5

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

See Also

`demo("solubility")` adds T-pH diagrams to the CO_2 and calcite example here. `demo("gold")` shows solubility calculations for Au in aqueous solutions with hydroxide, chloride, and hydrosulfide complexes. `equilibrate` calculates equilibrium chemical activities of species given a constant value of `loga.balance` (the logarithm of total activity of the conserved basis species).

Examples

```
## solubility of CO2 and calcite as a function of pH
opar <- par(mfrow = c(1, 2))
## set pH range and resolution, constant temperature and ionic strength
pH <- c(0, 14)
res <- 100
T <- 25
IS <- 0
## start with CO2 (not a dissociation reaction)
basis(c("carbon dioxide", "H2O", "O2", "H+"))
# ca. atmospheric PCO2
basis("CO2", -3.5)
species(c("CO2", "HCO3-", "CO3-2"))
a <- affinity(pH = c(pH, res), T = T, IS = IS)
s <- solubility(a)
# first plot total activity line
diagram(s, ylim = c(-10, 4), type = "loga.balance", lwd = 4, col = "green2")
# add activities of species
diagram(s, ylim=c(-10, 4), add = TRUE, dy = 1)
# add legend
lexpr <- as.expression(c("total", expr.species("CO2", state = "aq"),
```

```
  expr.species("HCO3-"), expr.species("CO3-2")))
legend("topleft", lty = c(1, 1:3), lwd = c(4, 2, 2, 2),
  col = c("green2", rep("black", 3)), legend = lexpr)
title(main = substitute("Solubility of"~what~"at"~T~degree*"C",
  list(what = expr.species("CO2"), T = T)), line = 1.5)
mtext("cf. Fig. 4.5 of Stumm and Morgan, 1996")
## now do calcite (a dissociation reaction)
basis(c("calcite", "Ca+2", "H2O", "O2", "H+"))
species(c("CO2", "HCO3-", "CO3-2"))
a <- affinity(pH = c(pH, res), T = T, IS = IS)
s <- solubility(a)
diagram(s, ylim = c(-10, 4), type = "loga.balance", lwd = 4, col = "green2")
diagram(s, add = TRUE, dy = 1)
legend("topright", lty = c(1, 1:3), lwd = c(4, 2, 2, 2),
  col = c("green2", rep("black", 3)), legend = lexpr)
title(main = substitute("Solubility of"~what~"at"~T~degree*"C",
  list(what = "calcite", T = T)))
mtext("cf. Fig. 4A of Manning et al., 2013")
par(opar)

## two ways to calculate pH-dependent solubility of calcite
## with ionic strength determination
## method 1: CO2 and carbonate species as formed species
basis(c("calcite", "Ca+2", "H2O", "O2", "H+"))
species(c("CO2", "HCO3-", "CO3-2"))
# ionic strength calculations don't converge below around pH=3
a <- affinity(pH = c(3, 14))
sa0 <- solubility(a)
saI <- solubility(a, find.IS = TRUE)
## method 2: CO2 and carbonate species as basis species
basis(c("calcite", "CO2", "H2O", "O2", "H+"))
species(c("Ca+2"))
m <- mosaic(c("CO2", "HCO3-", "CO3-2"), pH = c(3, 14))
sm0 <- solubility(m)
smI <- solubility(m, find.IS = TRUE)
## plot the results
plot(0, 0, xlab="pH", ylab="solubility, log mol", xlim = c(3, 14), ylim = c(-5, 2))
# method 1 with/without ionic strength
lines(a$vals[[1]], saI$loga.balance, lwd=5, col="lightblue")
lines(a$vals[[1]], sa0$loga.balance, lwd=5, col="pink")
# method 2 with/without ionic strength
lines(a$vals[[1]], smI$loga.balance, lty=2)
lines(a$vals[[1]], sm0$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) takes longer, 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, 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` (all species if that argument is missing)?
`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 operationally a combination of a `name` and `state`, which are columns of the thermodynamic database in `thermo$obigt`. The function operates on one or more character values of `species`. For each first match of `species` (optionally restricted to a `state` among '`aq`', '`cr`', '`gas`', '`liq`') to the name of a species or a formula or abbreviation in the thermodynamic database, a row is added to `thermo()$species`.

The data frame in `thermo()$species` holds the identifying characteristics of the species as well as the stoichiometric reaction coefficients for the formation of each of the species from the basis species, 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.

If `state` is `NULL` (the default), species in any state can be matched in the thermodynamic database. 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 phase 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 phase species are added.

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 states to which the species should be changed. If `species` is numeric and `delete` is `TRUE`, the rows representing these species are deleted from the dataframe; if the only argument is `delete` and it is `TRUE`, all the species are removed.

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")
# add, modify, delete species
species(c("CO2","NH3"))  # aqueous species
species(c("CO2","NH3"),"gas")  # gases
# delete the first couple of species
species(1:2,delete=TRUE)
# modify the logarithms of activities (actually
# fugacities) of the remaining species
species(1:2,c(-2,-5))
# set the species to aqueous
species(1:2,"aq")
# delete all the species (returns the existing species
# definition, then deletes the species)
sd <- 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
```

`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, action.unbalanced = "warn", 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 input and output units of T and P those of the user (TRUE) (see T.units), or are they Kelvin and bar (FALSE)?
action.unbalanced	character 'warn' or NULL, what action to take if unbalanced reaction is provided
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 equations-of-state functions for aqueous species (hkf), crystalline, gas, and liquid species (cgl) and liquid or supercritical H_2O (water).

T and P denote the temperature and pressure conditions for the calculations and should generally be of the same length, unless P is 'Psat' (the default; see water). Argument grid if present can be one of T or P to perform the computation of a T×P or P×T grid. The propertys 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}

Note that E and kT can only be calculated for aqueous species and only if the option (thermooptwater) 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.)

Depending on the units currently defined (E.units) the values of G, H, S and Cp are returned using calories or Joules as the unit of energy, but only if convert is TRUE. Likewise, the input values of T and P are interpreted to have the units specified through T.units and P.units, but setting convert to FALSE forces subcrt to treat them as Kelvin and bar, respectively.

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. Reactions that do not conserve elements are permitted; subcrt prints the missing composition needed to balance the reaction and produces a warning but computes anyway. Alternatively, if the basis species of a system were previously defined, and if the species being considered are within the compositional range of the basis species, an unbalanced reaction given in the arguments to subcrt will be balanced automatically, without altering the coefficients on the species identified in the arguments (unless perhaps they correspond to basis species), and without a warning. However, if a reaction is unbalanced and action.unbalanced is set to NULL, no warnings are generated and no attempt is made to balance the reaction.

Minerals with polymorphic transitions (denoted by having states 'cr' (lowest T phase), 'cr2', 'cr3' etc.) can be defined generically by 'cr' in the state argument with a character value for species. subcrt in this case simultaneously calculates the requested properties of all the phases of each such mineral (phase species) and, using the values of the transition temperatures calculated from those at P = 1 bar given in the thermodynamic database together with functions of the entropies and volumes of transitions (see dPdTtr), determines the stable phase of the mineral at any grid point and substitutes the properties of this phase at that point for further calculations. If individual phase species of minerals are specified (by 'cr', 'cr2' etc. in state), and exceed.Ttr is FALSE (the default), the Gibbs energies of these minerals are assigned values of NA at conditions beyond their transition temperature, where another of the phases is stable. If you set exceed.Ttr to TRUE to calculate the properties of mineral polymorphs (i.e., using 'cr') the function will identify the stable polymorph using the calculated Gibbs energies of the phase species instead of the tabulated transition temperatures. This is not generally advised, since the computed standard molal properties of a phase species lose their physical meaning beyond the transition temperatures of the phase.

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.

`subcrt` is modeled after the functionality of the SUPCRT92 package (Johnson et al., 1992). Certain features of SUPCRT92 are not available here, for example, calculations as a function of density of H_2O instead of pressure, or calculations of temperatures of univariant curves (i.e. for which `logK` is zero).

For calculations below 273.16 K, the pressure should be set to 1, as P_{SAT} is not defined in these conditions.

If `thermo()$opt$varP` 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

For `subcrt`, 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 phase 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 `demos("NaCl")` demonstrates one unexpected result.

`NA`s 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.

`NA`s 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`.

References

Anderson, G. M. and Crerar, D. A. (1993) *Thermodynamics in Geochemistry: The Equilibrium Model*, Oxford University Press. http://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. https://doi.org/10.1016/0098-3004(92)90029-Q

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. https://doi.org/10.1016/S0016-7037(97)00219-6

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. https://doi.org/10.1111/j.1472-4669.2007.00099.x

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

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. https://doi.org/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")
# calculating at different temperatures
subcrt("water", T=seq(0, 100, 10))
# calculating at even increments
subcrt("water", T=seq(500, 1000, length.out=10),
  P=seq(5000, 10000, length.out=10))
# calculating on a temperature-pressure grid
subcrt("water", T=c(500, 1000), P=c(5000, 10000), grid="P")
# to calculate only selected properties
subcrt("water", property=c("G", "E"))
# the properties of multiple species
subcrt(c("glucose", "ethanol", "CO2"))

## properties of reactions
subcrt(c("H2O", "H+", "K-feldspar", "kaolinite", "K+", "SiO2"),
  c(-1, -2, -2, 1, 2, 4))
subcrt(c("glucose", "ethanol", "CO2"), c(-1, 2, 2))
```

```
# to specify the states
subcrt(c("glucose", "ethanol", "CO2"), c(-1, 2, 2), c("aq", "aq", "gas"))

## auto balancing reactions
# the basis species must first be defined
basis(c("CO2", "H2O", "NH3", "H2S", "O2"))
subcrt(c("glucose", "ethanol"), c(-1, 3))
# a bug in CHNOSZ <0.9 caused the following
# to initiate an infinite loop
basis(c("H2O", "H2S", "O2", "H+"))
subcrt(c("HS-", "SO4-2"), c(-1, 1))
# because O2,aq is in the basis, this is a non-reaction
# (O2,aq to O2,aq)
subcrt("O2", 1, "aq")
# but this one auto-balances into a reaction
# (O2,aq to O2,gas)
subcrt("O2", 1, "gas")
# properties of a species and a formation
# reaction for that species
subcrt("C2H5OH")    # species
basis("CHNOS")
subcrt("C2H5OH", 1) # reaction

## mineral polymorphs
# properties of the stable polymorph
subcrt("pyrrhotite")
# properties of just the high-T phase
subcrt(c("pyrrhotite"), state="cr2")
# polymorphic transitions in a reaction
subcrt(c("pyrite", "pyrrhotite", "H2O", "H2S", "O2"), c(-1, 1, -1, 1, 0.5))

## these produce 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
## Not run:
## (also gives a warning)
subcrt("alanine", T=seq(0, 5000, by=1000))

## End(Not run)

## minerals with phase transitions
# compare calculated values of heat capacity of iron with
# values from Robie and Hemingway, 1995
T.units("K")
E.units("J")
# we set pressure here otherwise subcrt uses Psat (saturation
# vapor pressure of H2O above 100 degrees C) which can not 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")
E.units("cal")

## Skarn example after Johnson et al., 1992
P <- seq(500, 5000, 500)
# this is like the temperature specification used
# in the example by Johnson et al., 1992
# T <- seq(0, 1000, 100)
# we use this one to avoid warnings at 0 deg C, 5000 bar
T <- c(2, seq(100, 1000, 100))
s <- subcrt(c("andradite", "carbon dioxide", "H2S", "Cu+", "quartz",
  "calcite", "chalcopyrite", "H+", "H2O"),
  coeff=c(-1, -3, -4, -2, 3, 3, 2, 2, 3),
  state=c("cr", "g", "aq", "aq", "cr", "cr", "cr", "aq", "liq"),
  P=P, T=T, grid="P")
# The results are not identical to SUPCRT92, as CHNOSZ has updated
# parameters for species e.g. Cu+ from Shock et al., 1997.
# Check the calculated phase transitions for chalcopyrite
stopifnot(all.equal(s$polymorphs$chalcopyrite[1:11],
  c(1, 1, 1, 1, 1, 1, 2, 3, 3, 3, 3)))

## Standard Gibbs energy of reactions with HCN and
## formaldehyde, after Schulte and Shock, 1995 Fig. 1
rxn1 <- subcrt(c("formaldehyde","HCN","H2O","glycolic acid","NH3"),
  c(-1,-1,-2,1,1),P=300)
rxn2 <- subcrt(c("formaldehyde","HCN","H2O","glycine"),
  c(-1,-1,-1,1),P=300)
plot(x=rxn1$out$T,rxn1$out$G/1000,type="l",ylim=c(-40,-10),
  xlab=axis.label("T"),ylab=axis.label("DG0r","k"))
lines(rxn1$out$T,rxn2$out$G/1000)
# write the reactions on the plot
text(150, -14, describe.reaction(rxn1$reaction, iname=c(1,2,4)))
text(200, -35, describe.reaction(rxn2$reaction, iname=c(1,2)))
title(main=paste("Standard Gibbs energy of reactions",
  "after Schulte and Shock, 1995",sep="\n"))

## Calculation of chemical affinities
# after LaRowe and Helgeson, 2007, Fig. 3 (a): reduction of nicotinamide
# adenine dinucleotide (NAD) coupled to oxidation of glucose
# list the available NAD species
info("NAD ")
T <- seq(0, 120, 10)
# oxidation of glucose (C6H12O6)
basis(c("glucose", "H2O", "NH3", "CO2", "H+"), c(-3, 0, 999, -3, -7))
s <- subcrt(c("NAD(ox)-", "NAD(red)-2"), c(-12, 12), logact=c(0, 0), T=T)
# LH07's diagrams are shown per mole of electron (24 e- per 12 NAD)
A <- s$out$A/24/1000
plot(x=T, y=A, xlim=range(T), ylim=c(1.4, 5.4),
```

```
  xlab=axis.label("T"), ylab=axis.label("A", prefix="k"), type="l")
text("NAD(ox)-/NAD(red)-2 = 1", x=53, y=median(A), srt=21)
# different activity ratio
s <- subcrt(c("NAD(ox)-","NAD(red)-2"), c(-12, 12), logact=c(1, 0), T=T)
A <- s$out$A/24/1000
lines(x=T, y=A)
text("NAD(ox)-/NAD(red)-2 = 10", x=55, y=median(A), srt=24)
# different activity ratio
s <- subcrt(c("NAD(ox)-","NAD(red)-2"), c(-12, 12), logact=c(0, 1), T=T)
A <- s$out$A/24/1000
lines(x=T, y=A)
text("NAD(ox)-/NAD(red)-2 = 0.1", x=52, y=median(A), srt=18)
# print the reaction and chemical conditions on the plot
text(0, 5.3, describe.reaction(s$reaction, iname=c(1, 2)), adj=0)
text(0, 5.1, describe.basis(oneline=TRUE, ibasis=c(1, 2, 4, 5)), adj=0)
# label the plot
title(main=paste("Reduction of NAD coupled to oxidation of glucose",
 "after LaRowe and Helgeson, 2007", sep="\n"))

## 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)$out
# start plot with extra room on right
opar <- par(mar=c(5, 4, 4, 4))
# plot G
plot(sb$water$T, sb$water$G, ylim=c(-63000, -56000), xlab=axis.label("T"),
  ylab=axis.label("DG0"))
points(sb$`Na+`$T, sb$`Na+`$G, pch=2)
# add Cp
# change y-axis
par("usr"=c(par("usr")[1:2], -100, 25))
axis(4)
mtext(axis.label("Cp0"), side=4, line=3)
points(sb$water$T, sb$water$Cp, pch=16)
points(sb$`Na+`$T, sb$`Na+`$Cp, pch=17)
legend("topleft", pch=c(16, 1, 17, 2), legend=c("H2O Cp", "H2O G", "Na+ Cp", "Na+ G"))
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$G))]
# the boiling point increases with pressure
stopifnot(Tvap20 > Tvap2)
# more precisely, the calculated boiling points should be near the
# empirical values (digitized from Fig. 1 of Helgeson et al., 1998)
Tvap_2bar <- 156
Tvap_20bar <- 276
stopifnot(abs(Tvap2 - Tvap_2bar) < 6)
stopifnot(abs(Tvap20 - Tvap_20bar) < 25)
# those comparisons would fail if varP were FALSE (the default)
thermo("opt$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"))
# the logarithm of activity calculated for H2
# is equal to the one calculated from the equilibrium constant
# for H2O = H2 + 0.5O2
logK <- subcrt(c("oxygen","H2","H2O"), c(-0.5,-1,1), T=25)$out$logK
# the equilibrium value of logaH2
# (for logaH2O = 0 and logfO2 = -80)
(logaH2 <- -logK + 40)
stopifnot(all.equal(logaH2, b2$logact[5]))
# put O2 back in
b3 <- swap.basis("H2", "oxygen")
# we have returned to starting point
stopifnot(all.equal(b1$logact, b3$logact))

## demonstrating the interconvertibility between
## 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)
## note that basis.logact produces a named array
#stopifnot(all.equal(bl1, as.numeric(bl2)))

## swapping basis species while species are defined
## and using numeric species indices
basis("MgCHNOPS+")
# load some Mg-ATP species
species(c("MgATP-2", "MgHATP-", "MgH2ATP", "Mg2ATP"))
# swap in CO2(g) for CO2(aq)
iCO2g <- info("CO2", "gas")
swap.basis("CO2", iCO2g)
a1 <- affinity()
# swap in CH4(g) for CO2(g)
iCH4g <- info("CH4", "gas")
swap.basis(iCO2g, iCH4g)
```

```
a2 <- affinity()
# the equilibrium fugacity of CH4 is *very* low
# swap in CO2(aq) for CH4(g)
iCO2a <- info("CO2", "aq")
swap.basis(iCH4g, iCO2a)
a3 <- affinity()
# swapping the basis species didn't affect the affinities
# of the formation reactions of the species, since
# the chemical potentials of the elements were unchanged
stopifnot(all.equal(a1$values, a2$values))
stopifnot(all.equal(a1$values, a3$values))
```

`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 can be downloaded from `ftp://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"))
stopifnot(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
## we use 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 main data files provided with CHNOSZ, as `*.csv` files in the `extdata/thermo` and `extdata/OBIGT` directories of the package, are used to build the `thermo` object, which is described below.

Usage

```
reset()
obigt()
thermo(...)
```

Arguments

`...`	list, one or more arguments whose names correspond to the component() to modify

Format

- `thermo()$opt` List of computational settings. Square brackets indicate default values.

`cutoff`	numeric	Cutoff below which values are taken to be zero [1e-10] (see `makeup`)
`E.units`	character	The user's units of energy (['cal'] or 'J') (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 checkGHS produces a message (cal mol^{-1}) [100]
Cp.tol	numeric	Difference in Cp above which checkEOS produces a message (cal K^{-1} mol^{-1}) [1]
V.tol	numeric	Difference in V above which checkEOS 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]

- thermo()$element Dataframe containing the thermodynamic properties of elements taken from Cox et al., 1989 and Wagman et al., 1982. 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. Note the following database conventions:

  - The combination of name and state defines a species in thermo()$obigt. A species can not be duplicated (this is checked when running reset()).
  - English names of inorganic gases are used only for the gas state. The dissolved species is named with the chemical formula. Therefore, info("oxygen") refers to the gas, and info("O2") refers to the aqueous species.
  - Properties of most aqueous species (state = 'aq') are calculated using the revised Helgeson-Kirkham-Flowers (HKF) model (see hkf).
  - Properties of aqueous species with an NA value of Z (the final column of thermo()$obigt) are calculated using the Akinfiev-Diamond model (see AkDi).
  - Properties of most non-aqueous species (liquids, gases, and minerals) are calculated using a heat capacity polynomial expression with up to six terms (see cgl).
  - Properties of minerals with NA values of all heat capacity parameters are calculated using the Berman model (see berman).

'OrganoBioGeoTherm' is the name of a GUI program to use SUPCRT in Windows, produced in Harold C. Helgeson's Laboratory of Theoretical Geochemistry and Biogeochemistry at the University of California, Berkeley. The OBIGT database was originally developed for that program, and was the original basis for the database in CHNOSZ. There may be an additional meaning for the acronym: "One BIG Table" of thermodynamic data.

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 *Thermodynamic data in CHNOSZ* for a complete description of the sources of data. The original OBIGT database was influenced by the SUPCRT92 (Johnson et al., 1992) and SLOP98.DAT data files (Shock et al., 1998), and the references in those files are included here.

In order to represent thermodynamic data for minerals with phase transitions, the higher-temperature phases of these minerals are represented as phase species that have states denoted by 'cr2', 'cr3', etc. The standard molar thermodynamic properties at 25 °C and 1 bar (T_r and P_r) of the 'cr2' phase species of minerals were generated by first calculating those of the 'cr' (lowest-T) phase species at the transition temperature (T_{tr}) and 1 bar then taking account of the volume and entropy of transition (the latter can be retrieved by combining the former with the Clausius-Clapeyron equation and values of (dP/dT) of transitions taken from the SUPCRT92 data file) to calculate the standard molar entropy of the 'cr2' phase species at T_{tr}, and taking account of the enthalpy of transition (ΔH°, taken from the SUPCRT92 data file) to calculate the standard molar enthalpy of the 'cr2' phase species at T_{tr}. The standard molar properties of the 'cr2' phase species at T_{tr} and 1 bar calculated in this manner were combined with the equations-of-state parameters of the species to generate values of the standard molar properties at 25 °C and 1 bar. This process was repeated as necessary to generate the standard molar properties of phase species represented by 'cr3' and 'cr4', referencing at each iteration the previously calculated values of the standard molar properties of the lower-temperature phase species (i.e., 'cr2' and 'cr3'). A consequence of tabulating the standard molar thermodynamic properties of the phase species is that the values of (dP/dT) and ΔH° of phase transitions can be calculated using the equations of state and therefore do not need to be stored in the thermodynamic database. However, the transition temperatures (T_{tr}) generally can not be assessed by comparing the Gibbs energies of phase species and are tabulated in the database.

The identification of species and their standard molal thermodynamic properties at 25 °C and 1 bar are located in the first 12 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 (formatted as in SUPCRT92)
`G`	numeric	Standard molal Gibbs energy of formation from the elements (cal mol^{-1})
`H`	numeric	Standard molal enthalpy of formation from the elements (cal mol^{-1})
`S`	numeric	Standard molal entropy (cal mol^{-1} K^{-1})
`Cp`	numeric	Standard molal isobaric heat capacity (cal mol^{-1} K^{-1})
`V`	numeric	Standard molal volume (cm^3 mol^{-1})

The meanings of the remaining columns depend on the model used for a particular species (see database conventions above). The names of these columns are compounded from those of the parameters in the HKF equations of state and general heat capacity polynomial; for example, column 13 is named `a1.a`. Scaling of the values by orders of magnitude is adopted for some of the parameters, following common usage in the literature.

Columns 13-20 for aqueous species (parameters in the revised HKF equations of state):

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

Columns 13-20 for crystalline, gas and liquid species ($Cp = a + bT + cT^{-2} + dT^{-0.5} + eT^2 + fT^{\lambda}$).

`a`	numeric	a (cal K^{-1} mol^{-1})
`b`	numeric	$b \times 10^3$ (cal K^{-2} mol^{-1})
`c`	numeric	$c \times 10^{-5}$ (cal K mol^{-1})
`d`	numeric	d (cal K$^{-0.5}$ mol^{-1})
`e`	numeric	$e \times 10^5$ (cal K^{-3} mol^{-1})
`f`	numeric	f (cal K$^{-\lambda-1}$ mol^{-1})
`lambda`	numeric	λ (exponent on the f term)
`T`	numeric	Temperature of phase transition or upper temperature limit of validity of extrapolation (K)

Columns 13-20 for aqueous species using the Akinfiev-Diamond model. Note that the `c` column is used to store the ξ parameter, and that `Z` must be NA to activate the code for this model. The remaining columns 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` Dataframe of references to sources of 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 phase transitions of minerals, so individual phase species 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 `testthat/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

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. http://www.worldcat.org/oclc/18559968

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. https://doi.org/10.1016/0098-3004(92)90029-Q

Shock, E. L. et al. 1998 *SLOP98.dat* (computer data file). http://geopig.asu.edu/supcrt92_data/slop98.dat, accessed on 2005-11-05; moved to http://geopig.asu.edu/?q=tools.

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?
# nb. the .csv.xz files are loaded by default,
# and the .csv files have optional data that can be loaded with add.obigt()
```

```
dir(system.file("extdata/OBIGT", package = "CHNOSZ"))

## exploring thermo()$obigt
# what physical states there are
unique(thermo()$obigt$state)
# formulas of ten random species
n <- nrow(thermo()$obigt)
thermo()$obigt$formula[runif(10)*n]
```

`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 `array`s, 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 `i`th slice(s) in the `d`th 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 `d`th dimension using only the `i`th 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
x <- matrix(1:12,ncol=3)
# pay attention to the following when
# writing examples that test for identity!
identical(1*x,x)   # FALSE
# create two matrices that are multiples of the first
a <- 1*x
b <- 2*a
# these both have two dimensions of lengths 4 and 3
dim(a)  # 4 3
# combine them to make an array with three dimensions
c <- list2array(list(a,b))
# the third dimension has length 2
dim(c)  # 4 3 2
# the first slice of the third dimension == a
stopifnot(identical( slice(c,3), a ))
# the second slice of the third dimension == b
stopifnot(identical( slice(c,3,2), b ))
# 'slice' works just like the bracket operator
c11 <- slice(c,1)
c12 <- slice(c,1,2)
c21 <- slice(c,2,1)
c212 <- slice(c,2,1:2)
stopifnot(identical( c11, c[1,,] ))
stopifnot(identical( c12, c[2,,] ))
stopifnot(identical( c21, c[,1,] ))
stopifnot(identical( c212, c[,1:2,] ))
# let us replace part of the array
d <- slice(c,3,2,value=a)
# now the second slice of the third dimension == a
stopifnot(identical( slice(d,3,2), a ))
# and the sum across the third dimension == b
stopifnot(identical( dimSums(d,3), b ))
# taking the sum removes that dimension
dim(d)             # 4 3 2
dim(dimSums(d,1))  # 3 2
dim(dimSums(d,2))  # 4 2
dim(dimSums(d,3))  # 4 3

# working with an 'affinity' object

basis("CHNOS+")
species("alanine")
a1 <- affinity(O2=c(-80,-60))   # at 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)
stopifnot(all.equal(a1$values,a3$values))
```

`util.blast` *Functions to Work with BLAST Output Files*

Description

Read and filter BLAST tabular output files, make taxonomic identifications of the BLAST hits using gi numbers, write trimmed-down BLAST files.

Usage

```
read.blast(file, similarity = 30, evalue = 1e-5, max.hits = 1,
  min.length = NA, quiet = FALSE)
id.blast(blast, gi.taxid, taxid.names, min.taxon = 0,
  min.query = 0, min.phylum = 0, take.first = TRUE)
write.blast(blast, outfile)
def2gi(def)
```

Arguments

`file`	character, name of BLAST tabular output file
`similarity`	numeric, hits above this similarity score are kept
`evalue`	character, hits below this E value are kept
`max.hits`	numeric, up to this many hits are kept for each query sequence
`min.length`	numeric, hits with at least this alignment length are kept
`quiet`	logical, produce fewer messages?
`blast`	dataframe, BLAST table
`gi.taxid`	list, first component is sequence identifiers (gi numbers), second is taxon ids (taxids)
`taxid.names`	dataframe, with at least columns '`taxid`' (taxon id), '`phylum`' (name of phylum), '`species`' (name of species)
`min.taxon`	numeric, this taxon is kept if it makes up at least this fraction of total
`min.query`	numeric, query sequence is counted if a single phylum makes up this fraction of its hits
`min.phylum`	numeric, this phylum is kept if it makes up at least this fraction of total
`take.first`	logical, keep only first hit after all other filtering steps?
`outfile`	character, name of output file
`def`	character, FASTA defline(s)

Details

`read.blast` reads a BLAST (Altschul et al., 1997) tabular output `file` (such as generated using the -m 8 switch to the 'blastall' command), keeping only those hits with greater than or equal to `similarity` and less than or equal to `evalue` (expectation value). Furthermore, for each query sequence, only the top number of hits specified by `max.hits` are kept, and only hits with an alignment length of at least `min.length` are kept. One or more of these filters can be disabled by setting `similarity`, `evalue` and/or `max.hits` to NA.

`id.blast` takes a BLAST table (i.e., the output of `read.blast`) and finds the taxonomic ID, phylum and species name for each hit (subject sequence). The BLAST results are tied to taxids using `gi.taxid`, which is a list consisting of '`gi`' and '`taxid`' numeric vectors. Any subject sequence identifiers appearing in the BLAST file that do not match gi numbers in the `gi.taxid` list are dropped. The `taxid.names` dataframe lists the phylum and species names for each taxid.

`id.blast` furthermore performs three possible filtering steps, which are all disabled by default. If one or more of the arguments is set to a non-zero value, its operation is performed, in this order. Any taxon that does not initially make up at least the fraction of total hits given by `min.taxon` is removed. Any query sequence that does not have a single phylum making up at least the fraction of hits (for each query sequence) given by `min.query` is removed. Finally, any phylum that does not make up at least the fraction of total hits given by `min.phylum` is removed.

By default, for `take.first` equal to TRUE, `id.blast` performs a final filtering step (but `min.query` must be disabled). Only the first hit for each query sequence is kept.

`write.blast` takes a BLAST table (the output of `read.blast`) and writes to `outfile` a stripped-down BLAST file with empty values in the columns except for columns 1 (query sequence ID), 2 (hit sequence ID), 3 (similarity), 11 (E value). In the process, `def2gi` is used to extract the GI numbers for the hit sequences that are then kept in the second column. This function is used to reduce the size of the example BLAST files that are packaged with CHNOSZ (see the 'bison' section in `extdata`).

`def2gi` extracts the GI number from a FASTA defline.

Value

`read.blast` returns a dataframe with as many columns (12) as the BLAST file. `id.blast` returns a dataframe with columns `query`, `subject` (i.e., sequence id or gi number), `similarity`, `evalue`, `taxid`, `phylum` and `species`. `write.blast` `invisible`-y returns the results (that are also written to `outfile`).

References

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J. H., Zhang, Z., Miller, W. and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402. https://doi.org/doi:10.1093/nar/25.17.3389

Examples

```
## using def2gi
def <- "gi|218295810|ref|ZP_03496590.1|"
stopifnot(all.equal(def2gi(def), "218295810"))
```

```
## process some of the BLAST output for proteins
## from Bison Pool metagenome (JGI, 2007)
# read the file that connects taxids with the sequence identifier
tfile <- system.file("extdata/bison/gi.taxid.txt.xz", package="CHNOSZ")
gi.taxid <- scan(tfile, what=as.list(character(2)), flush=TRUE)
# read the file that connects names with the taxids
nfile <- system.file("extdata/refseq/taxid_names.csv.xz", package="CHNOSZ")
taxid.names <- read.csv(nfile)
# the BLAST files
sites <- c("N","S","R","Q","P")
bfile <- paste("extdata/bison/bison", sites, "_vs_refseq57.blastp.xz", sep="")
for(i in 1:5) {
  file <- system.file(bfile[i], package="CHNOSZ")
  # read the blast file, with default filtering settings
  bl <- read.blast(file)
  # process the blast file -- get taxon names
  ib <- id.blast(bl, gi.taxid, taxid.names, min.taxon=2^-7)
  # count each of the phyla
  bd <- as.matrix(sapply(unique(ib$phylum), function(x) (sum(x==ib$phylum))))
  colnames(bd) <- sites[i]
  # make a matrix -- each column for a different file
  if(i==1) bardata <- bd else {
    bardata <- merge(bardata, bd, all=TRUE, by="row.names")
    rownames(bardata) <- bardata$Row.names
    bardata <- bardata[,-1]
  }
}
# normalize the counts
bardata[is.na(bardata)] <- 0
bardata <- t(t(bardata)/colSums(bardata))
# make a bar chart
bp <- barplot(as.matrix(bardata), col=rainbow(nrow(bardata)),
  xlab="location", ylab="fractional abundance")
# add labels to the bars
names <- substr(row.names(bardata), 1, 3)
for(i in 1:5) {
  bd <- bardata[,i]
  ib <- bd!=0
  y <- (cumsum(bd) - bd/2)[ib]
  text(bp[i], y, names[ib])
}
title(main=paste("Phylum Classification of Protein Sequences",
  "in Part of the Bison Pool Metagenome", sep="\n"))
```

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

Description

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

Usage

```
thermo.refs(key=NULL, keep.duplicates=FALSE)
checkEOS(eos, state, prop, ret.diff = FALSE)
checkGHS(ghs, ret.diff = FALSE)
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()$obigt
state	character, physical state of species
prop	character, property of interest ('Cp' or 'V')
ret.diff	logical, return the difference between calculated and tabulated values?
ghs	dataframe, containing G, H and S, in the format of thermo()$obigt
file	character, path to a file
compound	character, name of compound(s) in group additivity calculation

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.

checkEOS compares heat capacity and volume calculated from equation-of-state parameters with reference (tabulated) values at 25 °C and 1 bar and prints a message and returns the calculated value if tolerance is exceeded. The Helgeson-Kirkham-Flowers equations of state parameters are in eos, which is a data frame with columns (and column names) in the same format as thermo$obigt. The property can be one of 'Cp' or V. The code only distinguishes between states of 'aq' and all others. The default tolerances, given in thermo()optCp.tol and thermo()optV.tol, are 1 cal/K.mol for Cp and 1 cm3/mol for V. If ret.diff is TRUE, the differences are returned irrespective of their values, and no messages are printed.

checkGHS compares G (standard molal Gibbs energy of formation from the elements) calculated from H (standard molal enthalpy of formation) and S (standard molal entropy) with reference (tabulated) values of G at 25 °C and 1 bar. A message is printed and the calculated difference is returned if it exceeds the value given in thermo()optG.tol, which has a default value of 100 cal/mol. The calculation requires that G, H and S, and the chemical formula of the species all be present.

check.obigt is a function to check self-consistency of each entry in the thermodynamic database, using checkEOS and checkGHS. 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). 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. https://doi.org/10.1016/S0016-7037(97)00345-1

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()$basis, prefix = "",
  molality = FALSE)
describe.basis(basis = thermo()$basis, ibasis = 1:nrow(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(ion = "K+", 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
basis	data frame, definition of basis species
ibasis	numeric, which basis species to include
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
ion	character, an ion

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, viz. (activity of the ion) / [(activity of H+) ^ (charge of the ion)].

See Also

`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(ibasis=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 since 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")
uniprot.aa(protein, start=NULL, stop=NULL)
```

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
`protein`	character, entry name for protein in UniProt

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).

uniprot.aa returns a data frame of amino acid composition, in the format of thermo()$protein, retrieved from the protein sequence if it is available from UniProt (http://uniprot.org). The protein argument corresponds to the 'Entry name' on the UniProt search pages.

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. nucleic.formula for an example of counting nucleobases in a DNA sequence.

Examples

```
## reading a protein FASTA file
# the path to the file
file <- system.file("extdata/fasta/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:
# download amino acid composition of a protein
# start at position 2 to remove the initiator methionine
aa <- uniprot.aa("ALAT1_HUMAN", start=2)
# add it to thermo()$protein
ip <- add.protein(aa)
# now it's possible to calculate some properties
protein.length(ip)
protein.formula(ip)
subcrt("ALAT1_HUMAN", c("cr", "aq"), c(-1, 1))
# the amino acid composition can be saved for future use
```

```
write.csv(aa, "saved.aa.csv", row.names=FALSE)
# in another R session, the protein can be loaded without using uniprot.aa()
aa <- read.csv("saved.aa.csv", as.is=TRUE)
add.protein(aa)

## count amino acids in a sequence
count.aa("GGSGG")
# warnings are issued for unrecognized characters
atest <- count.aa("WhatAmIMadeOf?")
# there are 3 "A" (alanine)
stopifnot(atest[, "A"]==3)

## 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)
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

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 cal 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`. 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. Units of cal mol^{-1} (`DG`, `DH`) and cal K^{-1} mol^{-1} (`S`) are assumed.

`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. https://doi.org/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). run.wjd uses the stoichiometric matrices created by i2A. protein.formula has an example of computing ZC for proteins compiled from the RefSeq database.

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"), S=water("S"))[1, ]
# that not quite equal to the value from water("G");
# probably using different entropies of the elements

## average oxidation states of carbon
stopifnot(ZC("CO2") == 4)
stopifnot(ZC("CH4") == -4)
stopifnot(ZC("CHNOSZ") == 7)
si <- info(info("LYSC_CHICK"))
stopifnot(si$formula == "C613H959N193O185S10")
stopifnot(all.equal(ZC(si$formula), 0.0163132137031))

## calculate the chemical formulas, then
## ZC of all of the proteins in CHNOSZ' 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.list *Functions to Work with Lists*

Description

Combine lists or perform arithmetic operations on elements of lists.

Usage

```
which.pmax(elts, na.rm = FALSE, pmin = FALSE)
```

Arguments

elts	list, numeric vectors for which to find maximum values (in parallel) (which.pmax).
na.rm	logical, remove missing values?
pmin	logical, find minimum values instead of maximum ones?

Details

which.pmax takes a list of equal-length numeric vectors (or objects that can be coerced to numeric) in elts and returns the index of the vector holding the maximum value at each position. If na.rm is TRUE, values of NA are removed; if pmin is TRUE the function finds locations of the minimum values instead.

---

util.matrix *Functions for Various Matrix Operations*

Description

Find rows of a matrix that form invertible (linearly independent) combinations.

Usage

```
invertible.combs(A, nmax=20)
```

Arguments

A	A matrix, with at least as many rows as columns.
nmax	The maximum number of rows to consider.

Details

Given a matrix A, with number of rows equal to or greater than the number of columns, return the combinations of row numbers that constitute invertible square matrices. Consider only the first nmax rows of the original matrix (to save time for large systems).

Examples

```
## what combinations of the 20 common amino acids have
## a linearly independent stoichiometry with five elements?
# the names of the amino acids
aanames <- aminoacids("")
# their species indices
iaa <- suppressMessages(info(aanames))
```

```
# the full stoichiometric matrix
A <- i2A(iaa)
# the invertible combinations
icA <- invertible.combs(A)
stopifnot(nrow(icA)==6067)
# that's a bit less than 40% of all possible combinations
nrow(icA) / ncol(combn(20, 5))
# count the occurrences of each amino acid
counts <- table(icA)
names(counts) <- aminoacids(1)
(sc <- sort(counts))
# the two sulfur-containing ones show up most frequently
stopifnot(tail(names(sc), 2)==c("C", "M"))
```

`util.misc` *Functions for Miscellaneous Tasks*

Description

Calculate dP/dT and temperature of phase 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 phase 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 phase 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 phase 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)$T, function(T) {
  # a very small T increment around the transition temperature
  T <- convert(c(T-0.01, T), "C")
  # use suppressMessages to make the output less crowded
  substuff <- suppressMessages(subcrt("iron", T=T, P=1))
  diff(substuff$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
stopifnot(identical(loga[1],loga[2]))
# the sum of activity of the residues is unity
stopifnot(isTRUE(all.equal(sum(10^loga * length),1)))
## now, 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
stopifnot(isTRUE(all.equal(loga[2]-loga[1],1)))
# but the activities of residues still add up to one
stopifnot(isTRUE(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 = 0.05, yfrac = 0.95, 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)
```

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, line 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

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 (see revisit for an example.)

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.

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. https://doi.org/10.1016/S0022-2836(05)80197-4

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 one- or three-letter abbreviations of amino acids; count nucleotides in nucleic acid sequences, calculate DNA and RNA complements of nucleic acid sequences.

Usage

```
aminoacids(nchar=1, which=NULL)
nucleic.formula(nucleic = NULL)
nucleic.complement(nucleic = NULL, type="DNA")
```

Arguments

nchar	numeric, 1 to return one-letter, 3 to return three-letter abbreviations for amino acids
which	character, which amino acids to name
nucleic	data frame, counts of nucleic-acid bases
type	character, target type of nucleic acid (DNA or RNA)

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).

`nucleic.formula` returns a string representation of the chemical formula for each nucleic-acid composition contained in `nucleic`. The names of the bases are indicated by the column names of `nucleic`. At present, the formula is computed as the sum of the chemical formulas of the bases themselves, with no contribution from polymerization (dehydration) or phosphorylation.

`nucleic.complement` calculates the complement of the base composition given in `nucleic`. `type` specifies the type of nucleic acid of the complement - 'DNA' (A, G, C, T) or 'RNA' (A, G, C, U).

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")
# the DNA complement of that sequence
DNA.comp <- nucleic.complement(bases)
# the RNA complement of the DNA complement
RNA.comp <- nucleic.complement(DNA.comp, type="RNA")
# the formula of the RNA complement (bases only)
nucleic.formula(RNA.comp)  # C40H42N32O11
```

---

util.test *Functions for Writing Tests*

Description

Functions modelled after the `expect_` functions in **testthat**.

Usage

```
maxdiff(x, y)
expect_maxdiff(object, expected, maxdiff = 0)
```

Arguments

`x`	numeric object
`y`	numeric object
`object`	numeric, object to test
`expected`	numeric, expected value
`maxdiff`	numeric, maximum pairwise difference between object and expected value

Details

`maxdiff` computes the maximum (absolute) pairwise difference between x and y, i.e. `max(abs(y - x))`.

`expect_maxdiff` tests that the maximum of the pairwise differences between two objects is less than the value of the argument `maxdiff`. The function uses `expect` to generate an expectation in the **testthat** framework.

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 (cal or J), and pressure (bar, 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	cal, cm^3 bar	calories, cm3bar
energy	cal, [none]	G, logK
oxidation potential	volt, [none]	Eh, pe
oxidation potential	volt, [none]	E0, logfO2

Examples

```
## examples using convert
# temperature (Kelvin) to degrees C
convert(273.15, "C")
# temperature (degrees C) to Kelvin
convert(100, "K")
# Gibbs energy (cal 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)
# calorie to/from joule
convert(10, "J")
convert(10, "cal")
# cm3bar to calories
convert(10, "calories")

### examples showing unit settings
# 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. https://doi.org/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
stopifnot(all(V.vapor > V.liquid))
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	cal mol^{-1}	*	*	NA
G	Gibbs energy	cal mol^{-1}	*	*	*
S	Entropy	cal K^{-1} mol^{-1}	*	*	NA
U	Internal energy	cal mol^{-1}	*	*	NA
H	Enthalpy	cal mol^{-1}	*	*	NA
Cv	Isochoric heat capacity	cal K^{-1} mol^{-1}	*	*	NA
Cp	Isobaric heat capacity	cal 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	cal 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. https://doi.org/10.1063/1.555853

Haar, L., Gallagher, J. S. and Kell, G. S. (1984) *NBS/NRC Steam Tables*. Hemisphere, Washington, D. C., 320 p. http://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. https://doi.org/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. https://doi.org/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. https://doi.org/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. https://doi.org/10.1016/0098-3004(92)90029-Q

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. https://doi.org/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. https://doi.org/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. https://doi.org/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. https://doi.org/10.1063/1.1461829

See Also

For calculating properties of reactions, `subcrt` coordinates the calculation of properties among `water` and `hkf` and `cgl` for other species.

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)
```

---

wjd *Gibbs Energy Minimization by Steepest Descent*

Description

Find the quantities of chemical species, subject to constant elemental bulk composition of the system, that minimize the Gibbs energy of the system.

Usage

```
wjd(
  A = matrix(c(
    1,2,2,0,0,1,0,0,0,1,
    0,0,0,1,2,1,1,0,0,0,
    0,0,1,0,0,0,1,1,2,1),ncol=3,
    dimnames=list(NULL,c("H","N","O"))),
  G0.RT = c(
```

```
    -10.021,-21.096,-37.986,-9.846,-28.653,
    -18.918,-28.032,-14.640,-30.594,-26.111),
  Y = c(0.1,0.35,0.5,0.1,0.35,0.1,0.1,0.1,0.1,0.1),
  P = 51,
  nlambda = 101,
  imax = 10,
  Gfrac = 1e-7
)
element.potentials(w, plot.it=FALSE, iplot=1:ncol(w$A))
is.near.equil(w, tol=0.01, quiet=FALSE)
guess(
  A = matrix(c(
    1,2,2,0,0,1,0,0,0,1,
    0,0,0,1,2,1,1,0,0,0,
    0,0,1,0,0,0,1,1,2,1),ncol=3,
    dimnames=list(NULL,c("H","N","O"))),
  B = c(2,1,1), method="stoich", minX = 0.001, iguess = 1, ic = NULL
)
run.wjd(ispecies, B = NULL, method = "stoich",
  Y = run.guess(ispecies, B, method), P=1, T=25, nlambda=101, imax = 10,
  Gfrac = 1e-7, tol = 0.01)
run.guess(ispecies, B = NULL, method = "stoich", iguess = NULL)
equil.potentials(w, tol=0.01, T=25)
```

Arguments

A	matrix, chemical formulas of the species (elements on columns)
G0.RT	numeric, the Gibbs energies / RT, at a single temperature (length equal to number of species)
Y	numeric, initial solution, a positive set of values (numbers of moles, length equal to number of species)
P	numeric, pressure in atmospheres
nlambda	numeric, number of values of fractional distance change (λ) tested at each step.
imax	numeric, maximum number of iterations
Gfrac	numeric, Gibbs energy change of system, as a fraction of total system energy in previous step, below which iterations will stop
w	list, output from wjd
plot.it	logical, make a plot?
iplot	numeric, which elements for which to make plots
tol	numeric, maximum difference in chemical potentials that counts as equilibrium
quiet	logical, don't output messages?
B	numeric, numbers of moles of the elements
method	character, method used for generating an initial solution
minX	numeric, minimum mole number for 'central' method

iguess	numeric, which guess to return
ic	numeric, which combination(s) of variable species to use (NULL for all)
ispecies	numeric, species indices (rownumbers of thermo$obigt)
T	numeric, temperature in °C

Details

wjd implements the steepest descent algorithm for Gibbs energy minimization in a closed system described by White et al., 1958. "Gibbs energy" (G) referred to here is the same as the "free energy" (F) used by White et al., 1958. wjd itself is independent of other functions or datasets in CHNOSZ, but run.wjd and run.guess are provided to make access to the thermodynamic database in CHNOSZ easier.

The default values of A, G0.RT, Y and P correspond to the example problem described by White et al., 1958 for gases in the H, N, O system at 3500 K. Note that for this, and for any other equilibrium problem that can be simulated using this function, the mole quantities in Y must all be positive numbers. Operationally, this vector defines not only the "initial solution" but also the bulk composition of the system; it is not possible to define the bulk composition using mole numbers of elements alone. The dimnames attribute in the default value for A gives the names of the elements; this attribute is not necessary for the function to operate, but is used in the examples below to help label the plots.

White et al., 1958 describe in detail the computation of the direction of steepest descent by means of Lagrange multipliers. They propose an iterative solution to the energy minimization problem, where at each step the mole numbers of species are recomputed and a new steepest descent direction calculated from there. However, the authors only give general guidelines for computing the value of λ, that is, the fraction of the total distance the system actually moves in the direction of steepest descent from the current point at each iteration. The two constraints given for determining the value of λ are that all mole numbers of species are positive, and the Gibbs energy of the system actually decreases (the minimum point is not passed). In the code described here, at each iteration the minimum value of lambda, not exceeding unity, that violates the first condition is computed directly (a value of one is assigned if the mole numbers remain positive through the entire range). With the default setting of nlambda, 101 values of λ at even intervals from 0 to this maximum permissible value are tested for the second condition at each iteration, and the highest conforming value is selected. If a value of 0 occurs, it means that the algorithm has reached an endpoint independently of the iteration and convergence tests (rho and Gfrac; see below). If this occurs, the value of nlambda might have to be increased depending on the user's needs.

The number of iterations is controlled by imax and Gfrac. The maximum number of iterations is set by imax; it can even be zero, though such a setting might only be useful in combination with element.potentials to characterize the initial state of a minimization problem. Within the limit of imax, the iterations continue until the magnitutde of the change in total Gibbs energy of the system, as a fraction of the system's energy in the previous iteration, is lower than the value specified in Gfrac. For the first example below, the default setting of Gfrac causes the algorithm to stop after six iterations.

Using the output of wjd, provided in the argument w, element.potentials calculates the chemical potentials of the elements in the system. It does so by combining the values of G0.RT of species with the inverses of stoichiometric matrices of combinations of species whose elemental compositions are linearly independent from each other. These possible combinations are constructed using the function invertible.combs. The value returned by element.potentials is a matrix, with each

column corresponding to a different element and each row to a different combination of species. The entries in the matrix are the chemical potentials of the elements divided by RT. If `plot.it` is set to TRUE, the chemical potentials of the elements are plotted as a function of species combination number, with as many plots as elements, unless `iplot` is set to another value (e.g. '`c(1,3)`' for only elements 1 and 3). In the first example below, the number of unique combinations of species is 120, but only 76 of these combinations provide stoichiometric independence.

There is no guarantee that the algorithm will converge on a global (or even be close to a local) minimum. However, some tests are available to help assess the likelihood that a solution is close to equilibrium. A necessary condition of equilibrium is that the chemical potentials of the elements be independent of the particular combination of species used to compute them. `is.near.equil` compares the chemical potentials for each element, computed using `element.potentials`, with the value of `tol`. If, for each element, the range of potentials/RT (difference between minimum and maximum) is less than `tol`, the result is TRUE, otherwise the function returns FALSE, and prints a message unless `quiet` is TRUE. The default value of `tol` corresponds to an energy of 0.01 * 1.9872 * 298.15 = ca. 6 cal/mol at 25 °C.

One of the constraints of the algorithm coded in `wjd` is that the initial solution, and all iterations, require positive (non-zero) numbers of moles of every species. Often, when investigating an equilibrium problem, the stoichiometric constraints are expressed most readily in terms of bulk composition – numbers of moles of each element. `guess` is a function to make initial guesses about the numbers of moles of all species in the system subject to the positivity constraints. Its system-specific arguments are the stoichiometric matrix `A` (as defined above for `wjd`) and the bulk composition vector `B`, giving the number of moles of each element, in the same order that the elements appear in `A`. The first `method` available in `guess` generates the 'central' solution of the system of linear equations using the `xranges` function from **limSolve**. The central solution is the mean of ranges of unknowns. The inequality constraint, or minimum number of moles of any species, is given by `minX`.

The second method for `guess` 'stoich' (and the default for `run.guess` and `run.wjd`) is to test successive combinations of species whose elemental compositions are linearly independent. The linearly independent combinations tested are all those from `invertible.combs` if `ic` is NULL, or only those identified by `ic`. Each combination is referred to as the 'variable' species; the moles of all 'other' species are set to a single value. This value is determined by the constraint that the greatest proportion, relative to the bulk composition in `B`, of any element contributed by all the 'other' species is equal to a value in `max.frac` (see code). The function tests nine hard-coded values of `max.frac` from 0.01 to 0.99, at each one solving for the moles of the 'variable' species that make up the difference in numbers of moles of elements. If the numbers of moles of all the 'variable' species is positive, the guess is accepted. The first accepted guess is returned if `iguess` is 1 (the default); other values of `iguess` indicate which guess to return. If `iguess` is NULL, all results are returned in a list, with non-successful guesses indicated by NA. In the first example below, of the 76 combinations of species whose elemental compositions are linearly independent, 32 yield guesses following this method.

`run.wjd` is a wrapper function to run `wjd`, provided the `ispecies` in the thermodynamic database (`thermo$obigt`), the chemical composition of the system in `B`, and the pressure `P` and temperature `T`; the latter are passed to `subcrt` (with `exceed.Ttr = TRUE`) to generate the values of `G0.RT` for `wjd`. Alternatively to `B`, the initial guess of numbers of moles of species can be provided in `Y`; otherwise as many combinations of `Y` as returned from `run.guess` are tested until one is found that `is.near.equil`. The function gives an error if none of the guesses in `Y` is near equilibrium, within the tolerance set by `tol`.

`run.guess` is a wrapper function to call `guess` using the stoichiometric matrix `A` built from the

ispecies indices in the thermodynamic database.

equil.potentials returns the average (colMeans) of element.potentials(w), or NULL if is.near.equil(w, tol=tol) is FALSE. The output of this function can be used as the emu argument for basis.logact to calculate the corresponding activities of the basis species.

Value

wjd returns a list with the problem definition and results: elements A, G0.RT, Y, and P are as supplied in the arguments; the results are in X (final mole numbers of species), F.Y (Gibbs energy of the system at initial conditions and after each iteration), lambda (value used for λ at each iteration), and elements (matrix of moles of elements at initial conditions and after each iteration; iterations on the columns and elements on the rows).

References

White, W. B., Johnson, S. M. and Dantzig, G. B. (1958) Chemical equilibrium in complex mixtures. *J. Chem. Phys.* **28**, 751–755. https://doi.org/10.1063/1.1744264

See Also

demo("wjd") for a longer example, and invertible.combs, used by element.potentials to find combinations of species that are compositionally independent.

Examples

```
## run the function with default settings to reproduce
## the example problem in White et al., 1958
w <- wjd()
# the mole fractions are very close to those shown in the
# last column of Table III in the paper
print(w$X)
# the Gibbs energy of the system decreased,
# and by a smaller amount, at each iteration
print(diff(w$F.Y))
# there are 76 unique combinations of species that can be
# used to calculate the chemical potentials of the elements
stopifnot(nrow(invertible.combs(w$A))==76)
# what the scatter in those chemical potentials looks like

ep <- element.potentials(w, plot.it=TRUE)
# the differences in chemical potentials / RT are all less than 0.01
is.near.equil(w)  # TRUE

## run the algorithm for only one iteration
w <- wjd(imax=1)
# the scatter in chemical potentials is much greater
ep <- element.potentials(w, plot.it=TRUE)
# and we're pretty far from equilibrium
is.near.equil(w)  # FALSE
```

```
## test all of the guesses of inititial mole quantities
## provided by guess() using default bulk composition (H2NO)
# 9 of them are not is.near.equil with the tolerance lowered to 0.0001
sapply( 1:32, function(i)
  is.near.equil(wjd(Y=guess(method = "stoich", iguess=i)), tol=0.0001) )

## using run.wjd(): 20 crystalline amino acids
iaa <- info(aminoacids(""), "cr")
# starting with one mole of each amino acid
w <- run.wjd(iaa, Y=rep(1, 20), imax=20)
# the following is TRUE (FALSE if tol is left at default)
is.near.equil(w, tol=0.012)
# in this assemblage, what are the amino acids
# in order of increasing abundance?
aminoacids()[order(w$X)]
# because the elements are redistributed among the species,
# the total number of moles of species does not remain constant
sum(w$X)  # <20
```

`yeast` *Composition, Localization, and Abundances of Proteins in Yeast*

Description

Retrieve the amino acid compositions of one or more proteins from *Saccharomyces cerevisiae* and get localizations and abundances reported by the YeastGFP project.

Usage

```
yeast.aa(protein = NULL)
yeastgfp(location, exclusive = TRUE)
```

Arguments

`protein`	character, name of protein
`location`	character, name of subcellular location (compartment)
`exclusive`	logical, report only proteins exclusively localized to a compartment?

Details

`yeast.aa` retrieves the amino acid composition(s) of the indicated proteins in *Saccharomyces cerevisiae*. The calculation depends on the data file `extdata/protein/Sce.csv.xz`, which contains the amino acid compositions of the proteins. The `protein` argument should be a vector or a list of vectors of one or more SGD IDs, Open Reading Frame (ORF) or gene names that are found in these files. The output data frame contains rows with NA compositions for names that are not matched.

yeastgfp returns the identities and abundances of proteins with the requested subcellular localization(s) (specified in location) using data from the YeastGFP project that is stored in extdata/abundance/yeastgfp.csv.x
If exclusive is FALSE, the function grabs all proteins that are localized to a compartment even if they are also localized to other compartments. If exclusive is TRUE (the default), only those proteins that are localized exclusively to the requested compartments are identified, unless there are no such proteins, then the non-exclusive localizations are used (applies to the 'bud' localization).

Value

For yeast.aa, a data frame, or list of data frames, containing the amino acid composition(s) of the specified protein(s) in the format of thermo$protein.

For yeastgfp, a list with elements named protein (names of proteins) and abundance (counts or concentrations without any conversion from the units in the data file). If location is NULL, yeastgfp returns the names of all known locations, and if the length of location is >1, the protein and abundance values are lists of the results for each location.

References

Boer, V. M., de Winde, J. H., Pronk, J. T. and Piper, M. D. W. (2003) The genome-wide transcriptional responses of *Saccharomyces cerevisiae* grown on glucose in aerobic chemostat cultures limited for carbon, nitrogen, phosphorus, or sulfur. *J. Biol. Chem.* **278**, 3265–3274. https://doi.org/10.1074/jbc.M209759200

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. https://doi.org/10.1074/jbc.M410573200

See Also

demos("yeastgfp")

Examples

```
# the first few names in UniProt for "aminotransferase yeast"
genes <- c("AATC", "ARO8", "BCA1", "AMPL", "BCA2", "ARO9")
# the corresponding ORF names
ORF <- c("YLR027C", "YGL202W", "YHR208W", "YKL103C", "YJR148W", "YHR137W")
# we only match two of them by gene name, but all by ORF name
aa <- yeast.aa(genes)
aa <- yeast.aa(ORF)
# what are their formulas and average oxidation states of carbon
protein.formula(aa)
ZC(protein.formula(aa))

## potential fields for overall protein compositions
## transcriptionally induced and repressed in aerobic
## and anaerobic carbon limitation
## (experiments of Tai et al., 2005)
# the activities of ammonium and sulfate used here
```

```
# are similar to the non-growth-limiting concentrations
# used by Boer et al., 2003
basis(c("glucose", "H2O", "NH4+", "hydrogen", "SO4-2", "H+"),
  c(-1, 0, -1.3, 999, -1.4, -7))
# the names of the experiments in TBD+05.csv
expt <- c("Clim.aerobic.down", "Clim.aerobic.up",
  "Clim.anaerobic.down", "Clim.anaerobic.up")
file <- system.file("extdata/abundance/TBD+05.csv", package="CHNOSZ")
dat <- read.csv(file, as.is=TRUE)
# yeast.aa: get the amino acid compositions
# aasum: average them together
for(thisexpt in expt) {
  p <- dat$protein[dat[, thisexpt]]
  aa <- yeast.aa(p)
  aa <- aasum(aa, average=TRUE, protein=thisexpt)
  add.protein(aa)
}
species(expt, "Sce")
a <- affinity(C6H12O6=c(-30, 0), H2=c(-20, 0))
d <- diagram(a, normalize=TRUE, fill=NULL)
title(main=paste("Formation potential of proteins associated with\n",
  "transcriptional response to carbon limitation in yeast"))
# the affinity of formation favors the proteins upregulated
# by carbon limitation at low chemical potentials of C6H12O6 ...
stopifnot(c(d$predominant[1,1], d$predominant[1,128])==grep("up", expt))
# ... and favors proteins downregulated by aerobic conditions
# at high hydrogen fugacities
stopifnot(c(d$predominant[128, 128], d$predominant[128, 1])==grep("down", expt))

## overall oxidation state of proteins exclusively localized
## to cytoplasm of S. cerevisiae with/without abundance weighting
y <- yeastgfp("cytoplasm")
aa <- yeast.aa(y$protein)
aaavg <- aasum(aa, average=TRUE)
ZC(protein.formula(aaavg))
# the average composition weighted by abundance
waaavg <- aasum(aa, abundance=y$abundance, average=TRUE)
ZC(protein.formula(waaavg))
```

Index