JEFFREY M. DICK

AN INTRODUCTION TO CHNOSZ

Overview

This document introduces the usage of CHNOSZ, a package for the R software environment. For more information on R, see "An Introduction to R" and the contributed documentation for R.

CHNOSZ has been developed since 2006 as a tool for thermodynamic calculations in geochemistry and compositional biology. The package provides functions and a thermodynamic database that can be used to calculate the stoichiometric and energetic properties of reactions involving minerals and inorganic and/or organic aqueous species. These functions also enable calculations of chemical affinities and metastable equilibrium distributions of proteins. A major feature of the package is the production of diagrams to visualize the effects of changing temperature, pressure, and activities of basis species on the potential for reactions among various species.

Installing and loading CHNOSZ

After starting R, install CHNOSZ by selecting the "Install packages from CRAN" or similar menu item in the R GUI or by using the following command:

Or, install the package from a package file, which you can download from CRAN or (for the development version) from R-Forge.

```
install.packages("CHNOSZ")
```

Then load the CHNOSZ package to make its functions available in your R session:

```
library(CHNOSZ)
```

Then load the `thermo` object, which contains the thermodynamic database and system settings for CHNOSZ:

```
data(thermo)
```

Note: Throughout this document, syntax highlighting is applied to the *input* of the code chunks. Double hash marks (`##`) precede the *output*, where black text denotes *results* and blue text is used for *messages*.

Getting help

After CHNOSZ is installed, type `help.start()` to browse the R help documents, then choose "Packages" followed by "CHNOSZ". That shows an index of the *manual* (help pages) for each function; many of the help pages include examples. There are also links to the *demos* (longer examples) and *vignettes* (more in-depth documentation; this document is a vignette).

Suggestions for accessing the documentation are indicated here with blue text. For example, read `` ?`CHNOSZ-package` `` to get an overview of the package and a list of features.

"`?`" is a shortcut to R's `help()` function. The command here is equivalent to `help("CHNOSZ-package")`.

Organization of major functions

CHNOSZ is made up of a set of functions and supporting datasets. The major components of the package are shown in the figure below, which is an updated version of the diagram in **?**. Rectangles and ellipses represent functions and datasets; bold text indicates primary functions.

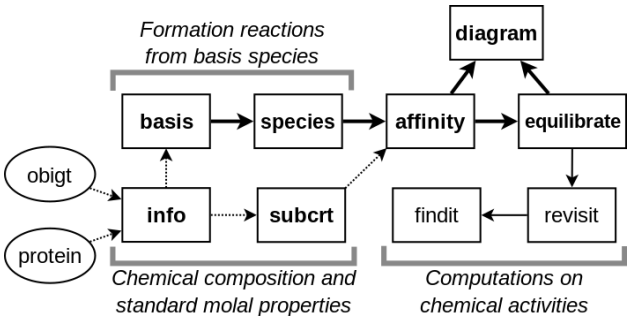


Figure 1: Structure of CHNOSZ.

Many functions in CHNOSZ have no side effects. That is, the function only returns a result; to use the result elsewhere, it can be assigned to a variable with `<-`. In this document, the names of these functions are set in green text (not applicable to the code chunks).

When they are mentioned, names of functions in the base and recommended packages of R are said to belong to R. Example: Use R's `plot()` to plot the data.

Major functions without side effects in CHNOSZ are:

- `info()`: search for species in the thermodynamic database;
- `subcrt()`: calculate the thermodynamic properties of species and reactions;
- `affinity()`: calculate the affinities of formation reactions using given chemical activities;
- `equilibrate()`: calculate the equilibrium chemical activities of the species of interest;
- `diagram()`: plot the results.

Some functions in CHNOSZ do have side effects: they modify

the `thermo` data object in the current R session. In this document, the names of these functions are set in red text (but not in the code chunks). Major functions with side effects are:

- `basis()`: set the basis species and their chemical activities;
- `species()`: set the species of interest and their (non-equilibrium) chemical activities;
- `data(thermo)`: reset the database, restoring all settings to their default values.

The following pseudocode shows a common sequence of commands. In actual usage, the `...` are replaced by arguments that define the chemical species and variables:

```
data(thermo)         ## initialize system settings
basis(...)
species(...)
a <- affinity(...)
e <- equilibrate(a)  ## optional
diagram(e)           ## or diagram(a)
data(thermo)         ## clear settings for next calculation
```

The basics

- Use `info()` to search the thermodynamic database.

```
info("aden ")
```

```
## [1] NA
```

```
info("adenine")
```

```
## [1] 1675
```

```
iadenine <- info("adenine")
```

```
info(iadenine)
```

```
##         name abbrv formula state        ref1  ref2      date     G     H     S
## 1675 adenine  <NA>  C5H5N5    aq LH06a [S07] LCT17 26.Jul.17 74770 31235 53.41
##         Cp    V    a1 a2     a3 a4    c1     c2   omega Z
## 1675 51.63 90.6 2.353  0 -17.75  0 48.54 -33180 -109300 0
```

- Use `thermo.refs()` to look up references.

```
thermo.refs(iadenine)
```

```
##       key                                    author year
## 99  LH06a          D. E. LaRowe and H. C. Helgeson 2006
## 152 LCT17 A. R. Lowe, J. S. Cox and P. R. Tremaine 2017
##                                    citation
## 99  Geochim. Cosmochim. Acta 70, 4680-4724
## 152   J. Chem. Thermodynamics 112, 129-145
##                                                 note
## 99  nucleic-acid bases, nucleosides, and nucleotides
## 152                           adenine HKF parameters
##                                             URL
## 99  https://doi.org/10.1016/j.gca.2006.04.010
## 152 https://doi.org/10.1016/j.jct.2017.04.005
```

- Use `subcrt()` to calculate standard molal thermodynamic properties.

```
subcrt("adenine", T = 100)
```

```
## $species
##         name formula state ispecies
## 1675 adenine  C5H5N5    aq     1675
##
## $out
## $out$adenine
##     T       P      rho     logK     G       H       S       V       Cp
## 1 100 1.01322 0.958393 -41.1609 70279 35542.8 66.2548 98.5326 82.4799
```

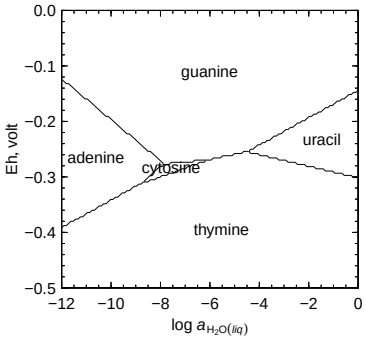


Figure 2: Nucleobase equal-activity diagram.

- Use `basis()` – `species()` – `affinity()` – `diagram()` (BSAD) to construct equal-activity (equipotential) diagrams.

```
basis("CHNOSe")
species(c("adenine", "cytosine", "guanine", "thymine", "uracil"))
a <- affinity(H2O = c(-12, -0), Eh = c(-0.5, 0), T = 100)
diagram(a)
```

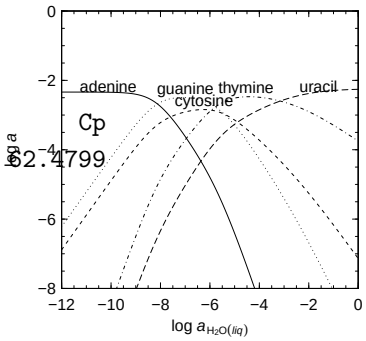


Figure 3: Activities of nucleobases in metastable equilibrium.

- Use `equilibrate()` to calculate equilibrium activities.

```
basis("e-", 3.6)
a <- affinity(H2O = c(-12, 0), T = 100)
e <- equilibrate(a)
diagram(e, ylim = c(-8, 0))
```

Thermodynamic database

An attempt has been made to provide a primary database (OBIGT) that has no major inconsistencies. As the database includes datasets from many sources, it can not be guaranteed to be fully internally consistent. For crucial problems, check not only the accuracy of entries in the database, but also the *suitability of the data* for your problem. If there are any doubts, consult the primary sources. Use `thermo.refs()` to show a list of references for the data; see also the vignette, *Thermodynamic data in CHNOSZ*, for more information.

The `info()` function

`info()` provides an interface to the thermodynamic database packaged with CHNOSZ. Suppose you are interested in the thermodynamic properties of aqueous methane. You can search for the species by name:

```
info("methane")
```

```
## [1] 859
```

Multiple entries exist for methane; the index of the `aq` (aqueous) species is returned by default. A second argument can be used to specify a different physical state:

```
info("methane", "gas")
```

```
## [1] 3307
```

Knowing that aqueous methane is species number 859 in the database, you can again use `info()` to retrieve the set of standard molal thermodynamic properties and equations of state parameters:

```
info(859)
```

```
##        name abbrv formula state       ref1 ref2      date     G      H  S    Cp
## 859 methane  <NA>     CH4    aq PS01 [S07] <NA> 04.Oct.00 -8140 -20930 21 60.23
##      V    a1    a2     a3     a4    c1    c2  omega Z
## 859 36 1.769 -1530 -67.88 114700 40.87 64500 -40000 0
```

This number can be used as an argument (`ispecies`) for other functions in CHNOSZ to uniquely identify any species; some commonly used functions also accept the species names. Liquid water is species number 1; it has NA entries in the database because specialized functions are used to compute its properties:

```
info(info("water"))
```

```
##    name abbrv formula state ref1 ref2      date  G  H  S Cp  V  a  b  c  d  e
## 1 water  <NA>     H2O   liq <NA> <NA> 25.Oct.06 NA NA NA NA NA NA NA NA NA NA
##    f lambda  T
## 1 NA     NA NA
```

Fuzzy searches

Calling `info()` with a string that does not exactly match the name of any species invokes a fuzzy search of the database:

```
info("acid")
```

```
## [1] NA
```

The message includes e.g. "uracil" and "metacinnabar" because their names have some similarity to the search term.

As "ribose" is the name of a species in the database, to find species with similar names, add an extra character to the search:

```
info(" ribose")
```

```
## [1] NA
```

The messages may be useful for browsing the database, but owing to their ambiguous results, these fuzzy searches return an `NA` value for the species index.

Counting elements, chemical formulas, `ZC()`

Continuing with the example of methane, let's look at its chemical formula:

```
info(859)$formula
```

```
## [1] "CH4"
```

We can use `makeup()` to count the elements in the formula, followed by `as.chemical.formula()` to rewrite the formula on one line:

```
makeup(859)
```

```
## C H
## 1 4
```

```
as.chemical.formula(makeup(859))
```

```
## [1] "CH4"
```

For organic species, a calculation of the average oxidation state of carbon (ZC) is possible given the species index, chemical formula, or elemental count:

```
ZC(859)
```

```
## [1] -4
```

```
ZC(info(859)$formula)
```

```
## [1] -4
```

```
ZC(makeup(859))
```

```
## [1] -4
```

Calculating thermodynamic properties

To calculate the standard molal properties of species and reactions, use `subcrt()`.

(**?**) If no reaction coefficients are given, `subcrt()` calculates the standard molal properties of individual species:

The inspiration for the name `subcrt()`, and the source of the Fortran subroutine used to calculate the thermodynamic properties of H2O, is SUPCRT (Johnson et al., 1992).

```
subcrt("water")
```

```
## $species
##    name formula state ispecies
## 1 water     H2O   liq        1
##
## $out
## $out$water
##          T         P      rho    logK        G        H       S       V      Cp
## 1    0.01   1.00000 0.999829 45.0353 -56289.5 -68767.7 15.1324 18.0183 18.2056
## 2   25.00   1.00000 0.997061 41.5525 -56687.7 -68316.8 16.7123 18.0683 18.0116
## 3   50.00   1.00000 0.988030 38.6328 -57123.9 -67866.5 18.1623 18.2335 18.0046
## 4   75.00   1.00000 0.974864 36.1544 -57594.9 -67416.1 19.5048 18.4797 18.0416
## 5  100.00   1.01322 0.958393 34.0270 -58098.4 -66963.8 20.7596 18.7973 18.1579
## 6  125.00   2.32014 0.939073 32.1832 -58631.7 -66507.3 21.9419 19.1840 18.3333
## 7  150.00   4.75717 0.917058 30.5718 -59193.3 -66045.6 23.0640 19.6446 18.5664
## 8  175.00   8.91805 0.892343 29.1531 -59781.4 -65576.6 24.1360 20.1887 18.8830
## 9  200.00  15.53650 0.864743 27.8960 -60394.5 -65098.0 25.1682 20.8330 19.3288
## 10 225.00  25.47860 0.833873 26.7753 -61031.2 -64605.9 26.1712 21.6042 19.9704
## 11 250.00  39.73649 0.799072 25.7711 -61690.3 -64095.0 27.1569 22.5452 20.9123
## 12 275.00  59.43125 0.759236 24.8670 -62370.7 -63557.5 28.1400 23.7281 22.3513
## 13 300.00  85.83784 0.712408 24.0494 -63071.1 -62980.9 29.1407 25.2878 24.7394
## 14 325.00 120.45757 0.654577 23.3072 -63790.8 -62341.4 30.1952 27.5219 29.4475
## 15 350.00 165.21129 0.574688 22.6310 -64528.9 -61575.6 31.3971 31.3478 43.5985
```

That uses the default temperature and pressure settings, i.e. equally spaced temperature intervals from 0 to 350 °C at Psat.

Psat is 1 bar below 100 °C, or the pressure of liquid-vapor saturation (i.e. boiling) at higher temperatures.

The columns in the output are temperature, pressure, density of water, logarithm of the equilibrium constant (only meaningful for reactions; see below), standard molal Gibbs energy and enthalpy of

formation from the elements, standard molal entropy, volume, and heat capacity.

A custom temperature-pressure grid can be specified. Here, we calculate the properties of H2O on a T, P grid in the supercritical region, with conditions grouped by pressure:

The corresponding units are °C (`T`), bar (`P`), g cm-3 (`rho`), cal mol-1 (`G` and `H`), cal K-1 mol-1 (`S` and `Cp`), and cm3 mol-1 (`V`).

See also `demo(density)`.

```
subcrt("water", T = c(400, 500, 600), P = c(200, 400, 600), grid = "P")$out$water
```

```
##     T   P       rho    logK        G        H       S        V      Cp
## 1 400 200 0.1005447 21.5271 -66306.4 -56639.3 39.0387 179.1760 27.4305
## 2 500 200 0.0677109 19.8868 -70353.7 -54819.8 41.5776 266.0607 14.0774
## 3 600 200 0.0550386 18.6704 -74593.1 -53540.0 43.1368 327.3197 11.9607
## 4 400 400 0.5236661 21.4119 -65951.4 -60454.4 32.8438  34.4021 37.5317
## 5 500 400 0.1779739 19.6639 -69565.0 -56252.5 38.7044 101.2238 24.9675
## 6 600 400 0.1238075 18.4124 -73562.4 -54361.7 41.0152 145.5098 15.4901
## 7 400 600 0.6124511 21.3631 -65801.3 -60832.4 32.0592  29.4149 25.8810
## 8 500 600 0.3384430 19.5660 -69218.6 -57694.2 36.3916  53.2296 32.4385
## 9 600 600 0.2071976 18.2784 -73027.2 -55195.1 39.4479  86.9469 19.3763
```

The additional operations (`$out$water`) are used to extract a specific part of the results; this can be used with e.g. R's `write.table()` or `plot()` for further processing:

```
substuff <- subcrt("water", T=seq(0,1000,100), P=c(NA, seq(1,500,1)), grid="T")
water <- substuff$out$water
plot(water$P, water$rho, type = "l")
```

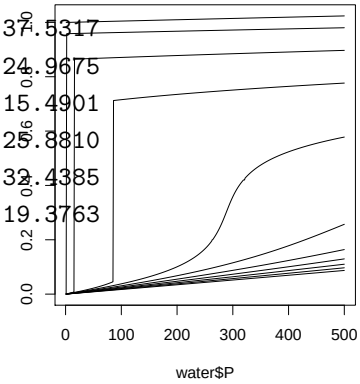


Figure 4: Isothermal contours of density (g cm⁻³) and pressure (bar) of water.

Changing units

The default units of temperature, pressure, and energy are °C, bar, and calories. The functions `T.units()`, `P.units()`, and `E.units()` can be used to change the units used by various functions in CHNOSZ. What is the Gibbs energy in J/mol of aqueous methane at 298.15 K and 0.1 MPa?

```
T.units("K")
P.units("MPa")
E.units("J")
subcrt("methane", T = 298.15, P = 0.1)$out$methane$G
```

```
## [1] -34057.8
```

A related function, `convert()`, can be used to convert given values between units. Let's convert the standard Gibbs energy of aqueous methane listed in the database from cal/mol to J/mol:

```
convert(info(info("methane"))$G, "J")
```

```
## [1] -34057.8
```

As expected, we get the same result from both operations.

Use `data(thermo)` to restore the units and all other settings for CHNOSZ to their defaults:

```
data(thermo)
```

Properties of reactions

Reaction definitions

To calculate the thermodynamic properties of reactions, give the names of species, the physical states (optional), and reaction coefficients as the arguments to `subcrt()`. Here we calculate properties for the dissolution of CO2:

Because of aqueous speciation, this doesn't give the *solubility* of CO2. For an example of a solubility calculation, see `demo(solubility)`, which is based on a figure in Manning et al. (2013).

```
subcrt(c("CO2", "CO2"), c("gas", "aq"), c(-1, 1), T = seq(0, 250, 50))
```

```
## $reaction
##      coeff           name formula state ispecies
## 3308    -1 carbon dioxide     CO2   gas     3308
## 1576     1            CO2     CO2    aq     1576
##
## $out
##        T        P      rho     logK       G         H         S       V      Cp
## 1   0.01  1.00000 0.999829 -1.10873 1385.80 -5907.949 -26.70130 23.6978 49.1458
## 2  50.00  1.00000 0.988030 -1.71865 2541.27 -3947.724 -20.08036 36.9843 34.6260
## 3 100.00  1.01322 0.958393 -2.00354 3420.89 -2271.995 -15.25623 41.5584 33.0446
## 4 150.00  4.75717 0.917058 -2.10770 4080.94  -596.111 -11.05290 44.5642 34.3421
## 5 200.00 15.53650 0.864743 -2.09920 4544.74  1228.815  -7.00814 47.9553 39.3857
## 6 250.00 39.73649 0.799072 -2.01184 4815.90  3513.101  -2.49026 54.3882 55.7180
```

In order to make a plot like Figure 18 of **?**, let's run more calculations and store the results. In addition to the reaction definition, we specify a greater number of temperature points than the default:

```
T <- seq(0, 350, 10)
CO2 <- subcrt(c("CO2", "CO2"), c("gas", "aq"), c(-1, 1), T = T)$out$logK
CO <- subcrt(c("CO", "CO"), c("gas", "aq"), c(-1, 1), T = T)$out$logK
CH4 <- subcrt(c("CH4", "CH4"), c("gas", "aq"), c(-1, 1), T = T)$out$logK
logK <- data.frame(T, CO2, CO, CH4)
```

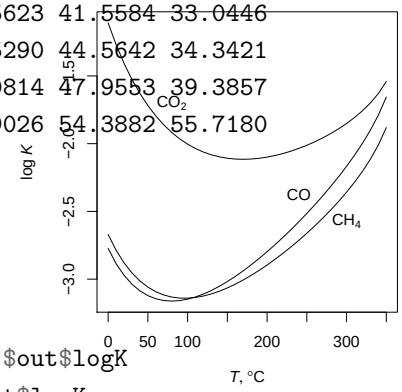


Figure 5: Calculated equilibrium constants for dissolution of CO₂, CO, and CH₄.

Now we can make the plot, using R's `matplot()`. Here, `axis.label()` and `expr.species()` are used to create formatted axis labels and chemical formulas:

```
matplot(logK[, 1], logK[, -1], type = "l", col = 1, lty = 1,
        xlab = axis.label("T"), ylab = axis.label("logK"))
text(80, -1.7, expr.species("CO2"))
text(240, -2.37, expr.species("CO"))
text(300, -2.57, expr.species("CH4"))
```

Unbalanced reactions

A balanced chemical reaction conserves mass. `subcrt()` won't stop you from running an unbalanced reaction, but it will give you a warning:

```
subcrt(c("CO2", "CH4"), c(-1, 1))
```

In other words, to balance the reaction, we should add 4 H to the left and 2 O to the right. That could be done manually be redefining the reaction with the appropriate species. There is another option: balancing the reaction automatically using basis species.

Setting the basis species

Basis species are a minimal number of chemical species that linearly combine to give the composition of any species in the system. The basis species are similar to thermodynamic components, but can include charged species. Basis species are used in CHNOSZ to automatically balance reactions; they are also required for making chemical activity diagrams.

Let's start with an example that doesn't work:

```
basis(c("CO2", "H2", "H2CO2"))
```

That set of species has a singular (non-invertible) stoichiometric matrix. An error would also result from either an underdetermined or overdetermined system. A valid set of basis species has an invertible stoichiometric matrix and the same number of species as elements:

```
basis(c("CO2", "H2", "H2O"))
```

```
##     C H O ispecies logact state
## CO2 1 0 2     1576      0    aq
## H2  0 2 0       64      0    aq
## H2O 0 2 1        1      0   liq
```

The composition of any species made up of C, H, and O can be represented by a single linear combination of these basis species.

Automatically balancing reactions

Methanogenic metabolism in reducing environments may proceed by acetoclastic or hydrogenotrophic processes. To consider reactions involving a charged species (acetate), let's define a basis with H+:

```
basis(c("CO2", "H2", "H2O", "H+"))
```

```
##     C H O Z ispecies logact state
## CO2 1 0 2 0     1576      0    aq
## H2  0 2 0 0       64      0    aq
## H2O 0 2 1 0        1      0   liq
## H+  0 1 0 1        3      0    aq
```

By identifying species *other than* the basis species, the reactions will be automatically balanced. This produces the balanced reaction for acetoclastic methanogenesis:

```
subcrt(c("acetate", "methane"), c(-1, 1))$reaction
```

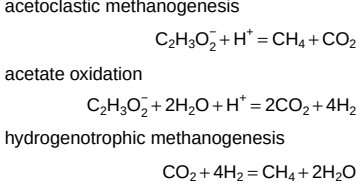
```
##      coeff    name formula state ispecies
## 1070    -1 acetate C2H3O2-    aq     1070
## 859      1 methane     CH4    aq      859
## 1576     1     CO2     CO2    aq     1576
## 3       -1      H+      H+    aq        3
```

We can similarly consider reactions for hydrogenotrophic methanogenesis as well as acetate oxidation (without production of methane):

```
acetate_oxidation <- subcrt("acetate", -1)
hydrogenotrophic <- subcrt("methane", 1)
acetoclastic <- subcrt(c("acetate", "methane"), c(-1, 1))
```

Use `describe.reaction()` to write the reactions on a plot:

```
plot(0, 0, type = "n", axes = FALSE, ann=FALSE, xlim=c(0, 5), ylim=c(5.2, -0.2))
text(0, 0, "acetoclastic methanogenesis", adj = 0)
text(5, 1, describe.reaction(acetoclastic$reaction), adj = 1)
text(0, 2, "acetate oxidation", adj = 0)
text(5, 3, describe.reaction(acetate_oxidation$reaction), adj = 1)
text(0, 4, "hydrogenotrophic methanogenesis", adj = 0)
text(5, 5, describe.reaction(hydrogenotrophic$reaction), adj = 1)
```



Chemical affinity

Usually, `subcrt()` returns only standard state thermodynamic properties.

The standard state adopted for H2O is unit activity of the pure component at any T and P. The standard state for aqueous species is unit activity of a hypothetical one molal solution referenced to infinite dilution at any T and P.

Thermodynamic models often consider a non-standard state (i.e. non-unit activity). The activities of basis species can be modified with `basis()`, and those of the other species using the `logact` argument in `subcrt()`.

Let us calculate the chemical affinity of acetoclastic methanogenesis.

The affinity is equal to the negative of the overall (non-standard) Gibbs energy change of the reaction.

We begin by changing the energy units to Joules. Then, we change the state of H2O and CO2 in the basis from `aq` (aqueous) to `gas`, and set the logarithm of fugacity of gaseous H2 and the pH, using values from `?`. The activity of acetate and fugacity of methane, as well as temperature and pressure, are set in the call to `subcrt()`:

```
E.units("J")
basis(c("CO2", "H2", "H2O", "H+"))
basis(c("CO2", "H2"), "gas")
basis(c("H2", "pH"), c(-3.92, 7.3))

subcrt(c("acetate", "methane"), c(-1, 1),
       c("aq", "gas"), logact = c(-3.4, -0.18), T = 55, P = 50)$out
```

```
##    T  P      rho    logK        G       H       S        V      Cp  logQ
## 1 55 50 0.987821 13.5984 -85429.6 18706.9 317.685 -40.1339 39.2144 10.52
##         A
## 1 19339.4
```

The new `A` column shows the affinity; the other columns are unaffected and still show the standard-state properties. Let's repeat the calculation for hydrogenotrophic methanogenesis.

```
subcrt("methane", 1, "gas", logact = -0.18, T = 55, P = 50)$out
```

```
##    T  P      rho   logK       G       H        S       V      Cp logQ       A
## 1 55 50 0.987821 18.828 -118284 -251807 -407.195 36.4746 33.4702 15.5 20907.8
```

Under the specified conditions, the affinities of hydrogenotrophic and acetoclastic methanogenesis are somewhat greater than and less than 20 kJ, respectively. This result matches Figure 4b in Mayumi et al. (2013) at unit fugacity of CO2.

The reproduction is not identical, owing to differences of thermodynamic data and of calculations of the effects of temperature and pressure.

We can go even further and reproduce their plot.

To make the code neater, we write a function that can run any of the reactions:

```
rxnfun <- function(coeffs) {
  subcrt(c("acetate", "methane"), coeffs,
         c("aq", "gas"), logact = c(-3.4, -0.18), T = 55, P = 50)$out
}
```

Now we're ready to calculate and plot the affinities. Here, we use R's `lapply()` to list the results at two values of logarithm of fugacity

of CO2. We insert an empty reaction to get a line at zero affinity. R's `do.call()` and `rbind()` are used to turn the list into a data frame that can be plotted with R's `matplot()`. There, we plot the negative affinities, equal to Gibbs energy, as shown in the plot of Mayumi et al. (2013).

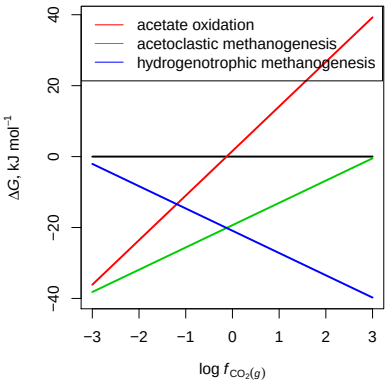


Figure 6: Gibbs energies of acetate oxidation and methanogenesis (after Mayumi et al., 2013).

```
Adat <- lapply(c(-3, 3), function(logfCO2) {
  basis("CO2", logfCO2)
  data.frame(logfCO2,
    rxnfun(c(0, 0))$A,
    rxnfun(c(-1, 0))$A,
    rxnfun(c(-1, 1))$A,
    rxnfun(c(0, 1))$A
  )
})
Adat <- do.call(rbind, Adat)
matplot(Adat[, 1], -Adat[, -1]/1000, type = "l", lty = 1, lwd = 2,
  xlab = axis.label("CO2"), ylab = axis.label("DG", prefix = "k"))
legend("topleft", c("acetate oxidation", "acetoclastic methanogenesis",
  "hydrogenotrophic methanogenesis"), lty = 1, col = 2:4)
```

Let's not forget to clear the system settings, which were modified by `basis()` and `E.units()`, before running other calculations:

```
data(thermo)
```

Using `affinity()`

`affinity()` offers calculations of chemical affinity of formation reactions over a configurable range of T, P, and activities of basis species.

Species of interest

By *formation reaction* is meant the stoichiometric requirements for formation of one mole of any species from the basis species. The `species()` function is used to set these *species of interest*. Let's consider the stoichiometry of some aqueous sulfur-bearing species. Here we use `basis()` with a keyword to load a preset basis definition.

Some available keywords are `CHNOS` (including CO2, H2O, NH3, H2S, and O2), `CHNOS+` (also including H+), and `CHNOSe` (including H+, and e- instead of O2). See `?basis` for more options.

```
basis("CHNOS+")
```

```
species(c("H2S", "HS-", "HSO4-", "SO4-2"))
```

```
##   CO2 H2O NH3 H2S O2 H+ ispecies logact state  name
## 1   0   0   0   1  0  0       67     -3    aq   H2S
## 2   0   0   0   1  0 -1       22     -3    aq   HS-
## 3   0   0   0   1  2 -1       25     -3    aq HSO4-
## 4   0   0   0   1  2 -2       24     -3    aq SO4-2
```

What is `SO42-`? Is it 1 S, 4 O, and 2 negative charges, or 1 S, 42 O, and 1 negative charge? The ambiguity of a digit that could belong to the coefficient for the following charge or to that for the preceding element is why formulas in CHNOSZ are written with the number of charges after the + or - symbol. `SO4-2` is unambiguously parsed as 1 S, 4 O and 2 negative charges.

Aqueous species are assigned default activities of 10-3 (`logact` is -3). Now, we can use `affinity()` to calculate the affinities of the formation reactions of each of the species. R's `unlist()` is used here to turn the list of values of affinity into a numeric object that can be printed in a couple of lines (note that the names correspond to `ispecies` above):

The values returned by `affinity()` are dimensionless, i.e. $A/(2.303RT)$.

```
unlist(affinity()$values)
```

```
##        67        22        25        24
##  -4.00000  -3.98775 -29.48836 -24.46748
```

The same result (but expressed in units of cal/mol or J/mol) could be obtained using `subcrt()`; however, `affinity()` has the advantage of being able to perform calculations on a grid of T, P, or activities

of basis species. Let's choose a set of variables commonly used in aqueous speciation diagrams: Eh and pH. To use Eh as a variable, the electron (e-) should be in the basis. To put the electron in there, we can use a different keyword (`basis("CHNOSe")`), or swap oxygen out of the existing basis:

```
swap.basis("O2", "e-")
```

```
##     C H N O S  Z ispecies   logact state
## CO2 1 0 0 2 0  0     1576 -3.00000    aq
## H2O 0 2 0 1 0  0        1  0.00000   liq
## NH3 0 3 1 0 0  0       66 -4.00000    aq
## H2S 0 2 0 0 1  0       67 -7.00000    aq
## e-  0 0 0 0 0 -1        2  6.22377    aq
## H+  0 1 0 0 0  1        3 -7.00000    aq
```

`swap.basis()` changed the basis species and recalculated their activities, but preserved the species of interest.

Now we can calculate the affinities on an Eh-pH grid:

```
a <- affinity(pH = c(0, 12), Eh = c(-0.5, 1))
```

That is, running `affinity()$values` again would give the same result.

Potential diagrams

Given values of affinity, the `diagram()` function uses the maximum affinity method to make a potential diagram (i.e. a Pourbaix diagram). Areas corresponding to Eh-pH conditions beyond the stability limits of water are colored slate gray. Another function, `water.lines()`, is used to draw lines at the water stability limits:

```
diagram(a, fill = "heat")
water.lines(a)
```

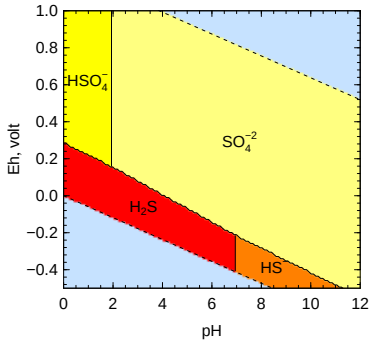


Figure 7: Aqueous sulfur species at 25 °C.

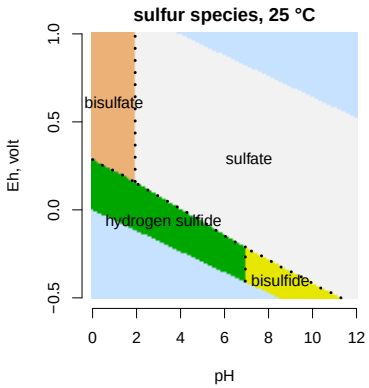


Figure 8: The same plot, with different colors and labels.

Note that the calculation of affinity implies a non-equilibrium reference state of equal activities of species (see above). Generally, then, `diagram()` gives a *potential diagram* because it shows regions of maximum affinity. In systems where equilibrium is attainable, it makes sense to call this a *predominance diagram*, showing regions of maximum activity.

The names of species that can be parsed as chemical formulas are formatted with subscripts and superscripts; if this is not desired, set `format.names = FALSE`. The default colors for diagrams shown on the screen use R's `heat.colors()` palette. Some arguments in `diagram()` can be used to control the color, labels, and lines, and title. The `tplot` argument turns off plot customizations used in CHNOSZ. Additional arguments are passed to R's plotting functions; here, we use `bty` to remove the box around the plot.

```
diagram(a, fill = "terrain", lwd = 3, lty = 3,
        names = c("hydrogen sulfide", "bisulfide", "bisulfate", "sulfate"),
        tplot = FALSE, main = "sulfur species, 25 °C", bty = "n")
```

Mineral stability diagrams often depict activity ratios, e.g. log (aCa+2/aH+2), on one or both axes. The variables used for potential calculations in CHNOSZ include only a single chemical activity, e.g. log aCa+2. However, you can set pH = 0 to generate diagrams that are geometrically equivalent to those calculated using activity ratios, and use `ratlab()` to make the axes labels for the ratios. See `demo(activity_ratios)` for some examples.

Mosaic diagrams

If sulfur is in the basis species, then we should consider that its speciation is sensitive to Eh and pH, as shown in the preceding diagram. Mosaic diagrams, which are often shown for oxide, sulfide, and carbonate minerals, account for speciation of the basis species. These diagrams are made by constructing individual diagrams for the possible basis species. The individual diagrams are then combined, each one contributing to the final diagram only in the range of stability of the corresponding basis species.

Let's use `mosaic()` to make a diagram for aqueous species and minerals in the Cu-S-Cl-H2O system, similar to Figure 5a of **?**. To know what aqueous copper chloride complexes are available in the database, we can use a fuzzy search:

```
info(" CuCl")
```

We wish to include chalcocite (Cu2S) in the system. This mineral undergoes phase transitions; to find out the temperatures of the phase transitions, we can also use `info()`:

```
info(info("chalcocite", c("cr", "cr2", "cr3")))$T
```

```
## [1]  376  717 1400
```

Those are temperatures in Kelvin (regardless of the `T.units()`); at 200 °C we should use the second phase.

Next we define the basis, and set the activities of the H2S and Cl- basis species. These represent the total activity of S and Cl in the system, which are distributed among the minerals and aqueous species. Three minerals and the aqueous copper chloride species are included:

```
basis(c("Cu", "H2S", "Cl-", "H2O", "H+", "e-"))
basis("H2S", -6)
```

```
basis("Cl-", -0.7)
species(c("copper", "tenorite"))
species("chalcocite", "cr2")
species(c("CuCl", "CuCl2-", "CuCl3-2", "CuCl+", "CuCl2", "CuCl3-", "CuCl4-2"))
```

We use `mosaic()` to generate and combine diagrams for each candidate basis species (H2S, HS-, HSO4-, or SO4-2) as a function of Eh and pH. The key argument is `bases`, which identifies the candidate basis species, starting with the one in the current basis. The other arguments, like those of `affinity()`, specify the ranges of the variables; `res` indicates the grid resolution to use for each variable (the default is 128). The first call to `diagram()` plots the species of interest; the second adds the predominance fields of the basis species. We turn off the gray coloring beyond the water stability limits (`limit.water`) but plot the red dotted lines using `water.lines()`:

```
T <- 200
res <- 200
bases <- c("H2S", "HS-", "HSO4-", "SO4-2")
m1 <- mosaic(bases, blend = TRUE, pH = c(0, 12, res), Eh=c(-1.2, 0.75, res), T=T)
diagram(m1$A.species, lwd = 2, fill = NA, limit.water = FALSE)
diagram(m1$A.bases, add = TRUE, col = "blue", col.names = "blue", lty = 2,
        limit.water = FALSE)
water.lines(m1$A.species, col = "red", lwd = 2, lty = 3)
```

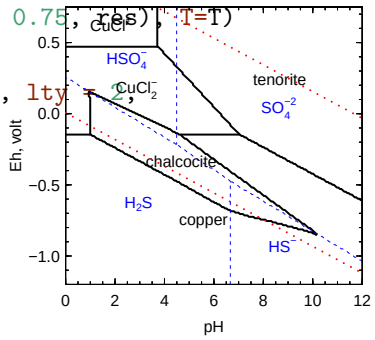


Figure 9: Copper minerals and aqueous complexes with chloride, 200 °C.

The argument `blend = TRUE` is used to combine the diagrams according to the equilibrium activities of the basis species by themselves (see below). The smooth transitions between basis species cause the appearance of curved lines on the plot. Without that argument, the diagrams would be combined using the dominant basis species, and all of the line segments would be straight.

We have seen the effects of speciation of S in the basis species. However, the choice of other basis species can also affect the diagram. For instance, we can use H2 or O2 in place of *e*-. To do that, let's write a function to swap those basis species and make a diagram. We use R's `do.call()` to construct the argument list for `mosaic()`; this way, the name of the `newvar` argument to our function indicates the chosen variable.

```
mosaicfun <- function(newvar, T = 200) {
  swap.basis("e-", names(newvar))
  if (names(newvar) == "O2") basis("O2", "gas")
  mosaicargs <- c(list(bases), blend=TRUE, pH=list(c(-2, 12, res)), newvar, T=T)
  m1 <- do.call(mosaic, mosaicargs)
  diagram(m1$A.species, lwd = 2, fill = rev(topo.colors(10)),
          limit.water = FALSE)
```

```
  diagram(m1$A.bases, add = TRUE, col = "blue", col.names = "blue", lty = 3,
          limit.water = FALSE)
  water.lines(m1$A.species, col = "red", lwd = 2, lty = 3)
  swap.basis(names(newvar), "e-")
}
par(mfrow = c(1, 3))
mosaicfun(list(Eh = c(-1, 1, res)))
mosaicfun(list(H2 = c(-30, 10, res)))
mosaicfun(list(O2 = c(-70, 5, res)))
```

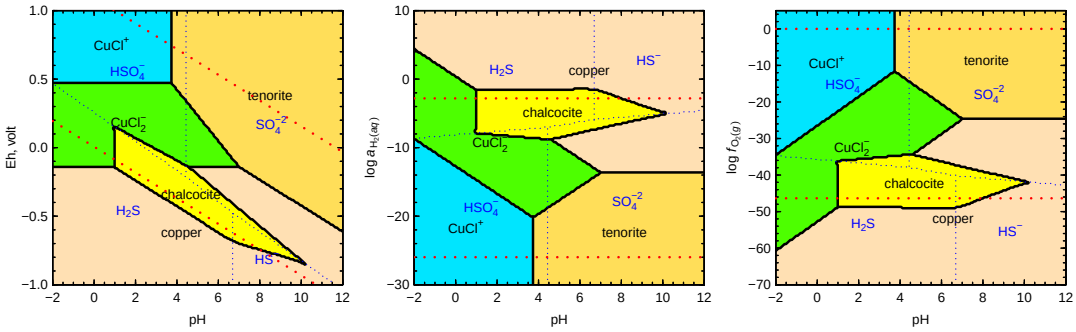


Figure 10: The same chemical system projected into different sets of basis species.

T, P, *activity transects*

Above, we used evenly-spaced grids of chemical activities of basis species; the ranges of variables were given by two or three values (minimum, maximum, and optionally resolution). `affinity()` can also perform calculations along a transect, i.e. a particular path along one or more variables. A transect is calculated when there are four or more values assigned to the variable(s). Let's use this feature to calculate affinities (negative Gibbs energies) of methanogenesis and biosynthetic reactions in a hydrothermal system.

Some results of mixing calculations for seawater and vent fluid from the Rainbow hydrothermal field, calculated using EQ3/6 by ?, are included in a data file in CHNOSZ. Reading the file with R's `read.csv()`, we set `check.names = FALSE` to preserve the `NH4+` column name (which is not a syntactically valid variable name):

```
file <- system.file("extdata/cpetc/SC10_Rainbow.csv", package = "CHNOSZ")
rb <- read.csv(file, check.names = FALSE)
```

We take a selection of the species from Shock and Canovas (2010) with activities equal to 10-6; methane is assigned an activity of 10-3. We will write the synthesis reactions of organic species in terms of these basis species:

The constant activity of methane is a simplification of the calculation reported by Shock and Canovas (2010). The code here could be expanded to vary the activity of methane.

```
basis(c("CO2", "H2", "NH4+", "H2O", "H2S", "H+"))
species("CH4", -3)
species(c("adenine", "cytosine", "aspartic acid", "deoxyribose",
          "methane", "leucine", "tryptophan", "n-nonanoic acid"), -6)
```

Now we can calculate affinities along the transect of changing temperature and activities of five basis species. Each variable is given as a named argument; `NH4+` must be quoted.

Using R's `lapply()` to run `convert()` for each species, we convert the affinity from dimensionless values ($A/(2.303RT)$) to cal/mol, then kcal/mol.

A shorter expression would use R's `do.call()` to construct the function call: `do.call(affinity, as.list(rb))`.

```
a <- affinity(T = rb$T, CO2 = rb$CO2, H2 = rb$H2,
              `NH4+` = rb$`NH4+`, H2S = rb$H2S, pH = rb$pH)
T <- convert(a$vals[[1]], "K")
a$values <- lapply(a$values, convert, "G", T)
a$values <- lapply(a$values, `*`, -0.001)
```

The target of the conversion is `G`, or free energy, from `logK`. That conversion requires temperature in Kelvin, which is obtained by conversion from °C. We finish with a negation (affinity is negative Gibbs energy) and scaling from cal to kcal.

Finally, we use `diagram()` to plot the results. Although only temperature is shown on the x axis, pH and the activities of CO2, H2, NH4+, and H2S are also varied according to the data in `rb`. By default, `diagram()` attempts to scale the affinities by dividing by the reaction coefficients of a shared basis species (in this case, CO2). To override that behavior, we set `balance = 1` to plot the affinities of the formation reactions as written (per mole of the product species).

```
diagram(a, balance = 1, ylim = c(-100, 100), ylab = axis.label("A", prefix="kJ"),
        col = rainbow(8), lwd = 2, bg = "slategray3")
abline(h = 0, lty = 2, lwd = 2)
```

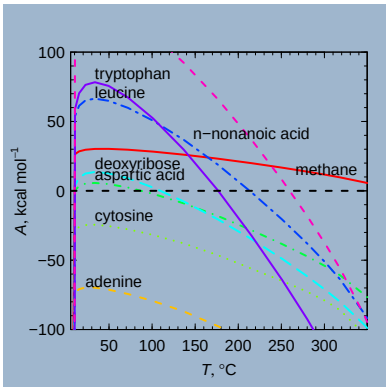


Figure 11: Affinities of organic synthesis in a hydrothermal system, after Shock and Canovas (2010).

When making line plots, `diagram()` automatically places the labels near the lines. The additional arguments `adj` and `dy` can be used to fine-tune the positions of the labels (they are used in a couple of examples below). If labeling of the lines is not desired, add e.g. `legend.x = "topright"` to make a legend instead, or `names = NULL` to prevent any plotting of the names.

Buffers

There is one other feature of `affinity()` that can be mentioned here. Can we go the other direction: calculate the activities of basis species from the activities of the species of interest? This question relates to the concept of chemical activity buffers. In CHNOSZ there are two ways to perform buffer calculations:

1. Assign the name of a buffer (listed in `thermo$buffer`) to the basis species:

- more versatile (multiple activities can be buffered, e.g. both S2 and O2 by pyrite-pyrrhotite-magnetite);
- the buffers are active in calculations of affinity of other species;
- use `mod.buffer()` to change or add buffers in `thermo$buffer`;
- `demo(buffer)` uses it for mineral buffers (solid lines).

2. Use the `what` argument of `diagram()` to solve for the activity of the indicated basis species:

- more convenient (the buffers come from the currently defined species of interest), but only a single basis species can be buffered, and it's not used in the calculation of affinity;
- `demo(buffer)` uses it for aqueous organic species as buffers (dotted and dashed lines).

As an example of method 1, let's look at the pyrite-pyrrhotite-magnetite (PPM) buffer at 300 °C.

For other examples, see `?buffer` and `demo(protbuff)` (hypothetical buffer made of proteins).

Without the buffer, the basis species have default activities of zero. Under these conditions, the minerals are not in equilibrium, as shown by their different affinities of formation:

```
basis(c("FeS2", "H2S", "O2", "H2O"))
species(c("pyrite", "magnetite"))
species("pyrrhotite", "cr2")
```

```
unlist(affinity(T = 300, P = 100)$values)
```

```
##     2113     2081     2118
##   0.0000 -50.6501 -20.6134
```

The affinity of formation of pyrite happens to be zero because it is identical to one of the selected basis species.

We use `mod.buffer()` to choose the `cr2` phase of pyrrhotite, which is stable at this temperature (see above for how to get this information for minerals with phase transitions). Then, we set up H2S and O2 to be buffered by PPM, and inspect their buffered activities:

```
mod.buffer("PPM", "pyrrhotite", "cr2")
```

```
basis(c("H2S", "O2"), c("PPM", "PPM"))
```

```
unlist(affinity(T = 300, P = 100, return.buffer = TRUE)[1:3])
```

```
##        H2S         O2       FeS2
##  -2.35581 -36.51522   0.00000
```

Et voilà! We have found log$a$H2S and logfO2 that are compatible with the coexistence of the three minerals. Under these conditions, the affinities of formation reactions of the minerals in the buffer are all equal to zero:

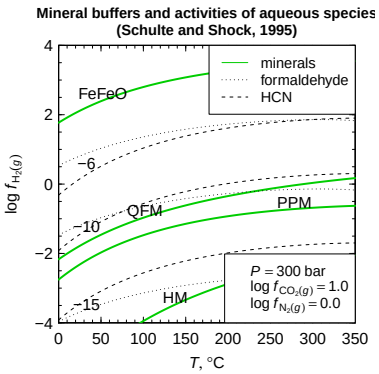


Figure 12: Values of logf_{H₂} corresponding to mineral buffers or to given activities of aqueous species.

```
unlist(affinity(T = 300, P = 100)$values)
## 2031 1999 2036
##    0    0    0
```

Another example, based on Figure 6 of **?**, is given in `demo(buffer)`. Here, values of log$f$H2 buffered by minerals or set by equilibrium with given activities of aqueous species are calculated using the two methods:

```
demo(buffer)
```

Equilibration

Above we considered this kind of question: for equal activities of species, what are the affinities of their formation reactions from basis species? Turning the question around, we would like to know: for equal affinities, what are the activities of species? This is the question of equilibration.

Before presenting some examples, it is helpful to know about the limitations of the functions. CHNOSZ does not take account of all possible reactions in the speciation of a system. Instead, it assumes that the total activity of species in the system is set by the activity of *one* basis species.

This balanced, or conserved, basis species must be present (with a positive or negative coefficient) in the formation reactions of all species considered.

For a given total activity of the balanced basis species, activities of the species can be found such that the affinities of the formation reactions are all equal. This is an example of metastable equilibrium. With additional constraints, the affinities of the formation reactions are not only equal to each other, but equal to zero. This is total equilibrium. An example of total equilibrium was given above for the PPM buffer. In contrast, models for systems of organic and biomolecules often involve metastable equilibrium constraints.

When activity coefficients are assumed to be zero, activities are equal to concentration and we can refer to "total activity". If the ionic strength is specified, `nonideal()` (see below) can be used to calculate activity coefficients.

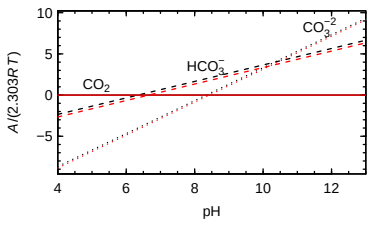
Getting from affinity to equilibrium

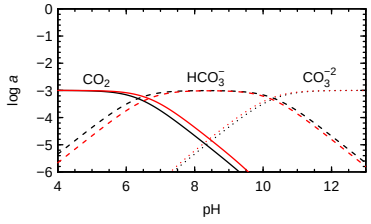
The `equilibrate()` function in CHNOSZ automatically chooses between two methods for calculating equilibrium.

The method based on the Boltzmann equation is fast, but is applicable only to systems where the coefficient on the balanced basis species in each of the formation reactions is one. The reaction-matrix method is slower, but can be applied to systems were the balanced basis species has reaction coefficients other than one.

The distribution of aqueous carbonate species as a function of pH (a type of Bjerrum plot) is a classic example of an equilibrium calculation. We can begin by plotting the affinities of formation, for equal

For more information, see the vignette *Equilibrium in CHNOSZ*.





activities of the species, calculated at 25 °C and 150 °C. Here, CO2 is in the basis, so it has zero affinity, which is greater than the affinities of HCO3- and CO3-2 at low pH. To avoid overplotting the lines, we offset the species labels in the y direction using the `dy` argument:

```
par(mfrow = c(3, 1))
basis("CHNOS+")
species(c("CO2", "HCO3-", "CO3-2"))
a25 <- affinity(pH = c(4, 13))
a150 <- affinity(pH = c(4, 13), T = 150)
diagram(a25, dy = 0.4)
diagram(a150, add = TRUE, col = "red")
```

Now we use `equilibrate()` to calculate the activities of species. Our balancing constraint is that the total activity of C is 10-3. This shows a hypothetical metastable equilibrium; we know that for true equilibrium the total activity of C is affected by pH.

```
e25 <- equilibrate(a25, loga.balance = -3)
e150 <- equilibrate(a150, loga.balance = -3)
diagram(e25, ylim = c(-6, 0), dy = 0.15)
diagram(e150, add = TRUE, col = "red")
```

To display the species distribution, or degree of formation, add `alpha = TRUE` to the argument list:

```
diagram(e25, alpha = TRUE, dy = -0.25)
diagram(e150, alpha = TRUE, add = TRUE, col = "red")
```

The possible reactions between species are all balanced on 1 C. Therefore, although pH alters the total activity of C, in a system with ideal mixing the total activity of C doesn't affect the relative activities of these species.

See `demo(solubility)` for calculations of the total activity of C in this ideal system; uncomment a line in the demo to run calculations for CO2 instead of calcite.

Groups of species

Sometimes it is helpful to look at the summed activities of species as groups on species distribution diagrams. The `groups` argument of `diagram()` can be used to sum the activities of species.

To demonstrate this feature, let's consider the distribution of carbon among organic and inorganic species in the hydrothermal mixing scenario described by `?`. First we define the basis and add two inorganic species. The `index.return = TRUE` argument tells `info()` to return the index (number) of the species in the current species definition; these indices are saved for use below:

```
basis("CHNOS+")
ii <- species(c("CO2", "HCO3-"), index.return = TRUE)
```

Next, we add each group of organic species: C1–C8 alcohols, C3–C8 ketones, C2–C12 carboxylic acids and their corresponding anions, and C2–C8 alkenes. To do this, we provide `info()` with a set of `ispecies` values that identify these species. In the database, the species in each group are ordered by carbon number, so we construct a sequence from the starting to ending `ispecies` for each group using R's `seq()` function, wrapped by the `seq2()` function we write here to make the code shorter:

```
seq2 <- function(x) seq(x[1], x[2])
ia <- species(seq2(info(c("methanol", "octanol"))), index.return = TRUE)
ik <- species(seq2(info(c("acetone", "2-octanone"))), index.return = TRUE)
ic <- species(seq2(info(c("acetic acid","n-dodecanoic acid"))),index.return=TRUE)
ica <- species(seq2(info(c("acetate", "n-dodecanoate"))), index.return = TRUE)
ie <- species(seq2(info(c("ethylene", "octene"))), index.return = TRUE)
```

Now we read two data files that contain values of logfO2 and pH as a function of temperature, digitized from Figure 5 of Shock and Schulte (1998).

The specific values are for calculations with vent fluids initially set by the fayalite-magnetite-quartz buffer minus 1/2 log$f$O2 (FMQ - 1/2).

These values were calculated using a speciation and mixing model that is not available in CHNOSZ; however, we can use these intermediate values as input to the "downstream" calculations that are available in CHNOSZ. Because of the noise introduced by digitization of the figure, we smooth the data using R's `smooth.spline()`; the lower T limit reflects the absence of data below this temperature in the figure for log$f$O2.

```
O2dat <- read.csv(system.file(
  "extdata/cpetc/SS98_Fig5a.csv", package = "CHNOSZ"))
pHdat <- read.csv(system.file(
  "extdata/cpetc/SS98_Fig5b.csv", package = "CHNOSZ"))
T <- seq(8, 350)
O2 <- predict(smooth.spline(O2dat$T, O2dat$logfO2), T)$y
pH <- predict(smooth.spline(pHdat$T, pHdat$pH), T)$y
```

We are ready to calculate affinities and equilibrium activities of the species. This calculation utilizes the transect mode of `affinity()` (see above). The call to `equilibrate()` runs with the default balance (in this case, CO2), with a log activity set to -2.5.

Actually, the total concentration of carbon depends on the mixing ratio, ranging from about 10-2.2 (seawater) to 10-2.6 (vent fluid). A setting to vary the activity of the balanced basis species is not yet implemented in CHNOSZ, so a single value is used here.

```
a <- affinity(T = T, O2 = O2, pH = pH)
e <- equilibrate(a, loga.balance = -2.5)
```

At last we come to the diagram. The groups are identified by the current species numbers in a list; the elements of the list are given names that will appear on the diagram. When summing the activities

of species in the groups, each activity is multiplied first by the balance coefficient on that species. Therefore, the total activity is that of a basis species (or of an element that is present only in that basis species, like carbon in this example).

```
par(mfrow = c(1, 3))
groups <- list(inorganic = ii, alcohols = ia, ketones = ik,
               `carboxylic acids` = c(ic, ica), alkenes = ie)
diagram(e, alpha = TRUE, groups = groups, col = 1:5)
```

That makes a diagram that is similar to Figure 6b of Shock and Schulte (1998).

Some differences from the original diagrams could be caused by the sensitivity of the calculations to log$f$O2, for which the values we use here may carry artifacts introduced by digitization of their plot.

It is also possible to plot the distribution of species within individual groups, such as alcohols and ketones. We do this by setting the names and line types for the *other* species to values that prevent them from being plotted:

```
# plot only alcohols
names <- within(species(), name[-ia] <- "")$name
lty <- ifelse(names == "", 0, 1)
diagram(e, alpha = TRUE, ylim = c(0, 0.32), lty = lty, names = names)
# plot only ketones
names <- within(species(), name[-ik] <- "")$name
lty <- ifelse(names == "", 0, 1)
diagram(e, alpha = TRUE, ylim = c(0, 0.16), lty = lty, names = names)
```

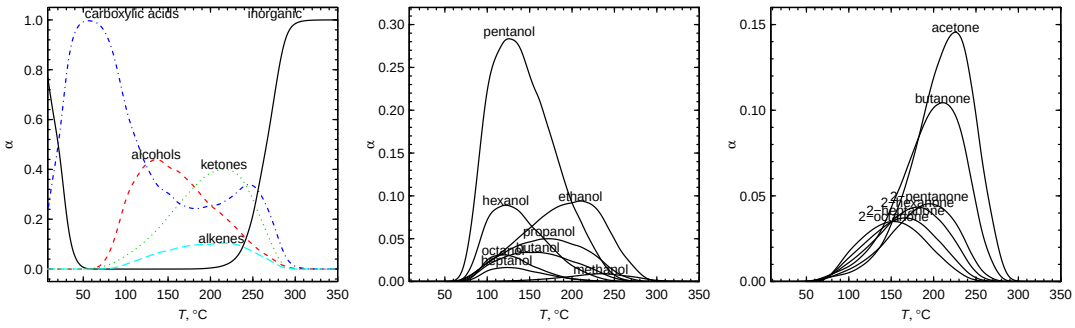


Figure 14: Distribution of inorganic and groups of organic species (left plot) and of alcohols and ketones (middle and right plots) as a function of T, pH, and logf_{O₂}.

Balancing differently

How about the choice between balancing constraints? Be default, `equilibrate()` and `diagram()` balance reactions on the first basis species that is present in each of the species of interest. Let's look at some amino acids in a hypothetical metastable equilibrium. This calculation is loosely based on one described by **?** for five amino acids.

Here we include 20 proteinogenic amino acids, whose names are returned by `aminoacids("")`. We use `ZC.col()` to generate colors based on the average oxidation state of carbon of the amino acids (red and blue for relatively reduced and oxidized).

```
basis("CHNOS")
basis("CO2", "gas")
swap.basis("NH3", "N2")
species(aminoacids(""))
a <- affinity(O2 = c(-50, -25, 200), CO2 = c(-10, 15, 200), T = 250, P = 265)
aa.ZC <- ZC(info(aminoacids("")))
col <- ZC.col(aa.ZC)
```

To make plots using different balance constraints, let's write a function that sets the `balance` argument of `diagram()` and adds a title to the plot. The first plot is the most similar to Figure 4 of Shock (1990), except for the absence of alanine (probably due to different thermodynamic data) and the presence of some other amino acids. There, we set `balance = 1`, which indicates that moles of species are conserved; this is equivalent to balancing on the amino acid backbone. In the remaining plots, the balance is set to each of the basis species in turn (except for O2), then on volume. `expr.species()` together with R's `substitute()` is used to make titles that include formatted chemical formulas:

```
aafun <- function(balance) {
  diagram(a, balance = balance, fill = col)
  blab <- expr.species(balance)
  title(main = substitute("balanced on" ~ b, list(b = blab)))
}
par(mfrow = c(1, 5))
lapply(c("1", "CO2", "H2O", "N2", "volume"), aafun)
```

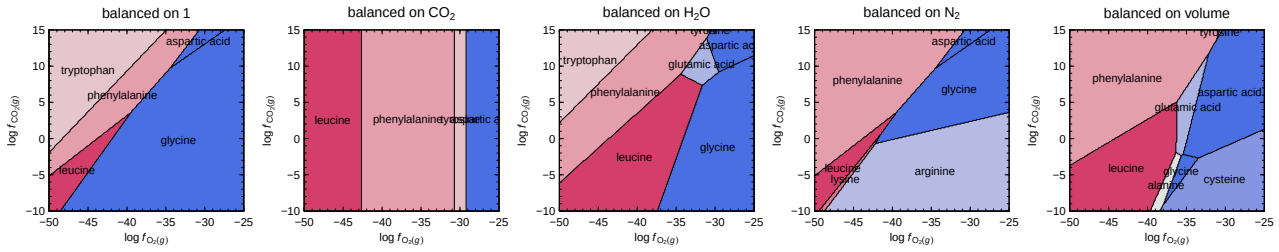


Figure 15: Plots of maximum affinity at 250 °C and 265 bar using different reaction balances for 20 amino acids.

There are some broad similarities—increasing logfO2 favors more oxidized amino acids—but also substantial differences. It is interesting that there is more "going on" in the middle part of the diagram showing volume conservation.

Caveat lector. These plots demonstrate some possibilities in CHNOSZ and are not necessarily realistic portrayals of this system. It does seem odd to balance on a fugacious component like O2 or H2O. Unlike different choices of basis species, which are thermodynamically equivalent (see above), the choice of balance reflects extra-thermodynamic factors. For instance, the widespread rule of thumb for balancing mineral reactions on a chemical component is unrealistic for processes where volume is conserved (**?**). While choosing an inappropriate balance leads to infeasible models, consideration of the different possibilities might give insight into the conditions affecting the dynamics of some systems.

Activity coefficients

For calculating activity coefficients of charged species, `nonideal()` uses the extended Debye–Hückel equation as parameterized by **?** for NaCl-dominated solutions to high pressure and temperature, or optionally using parameters described in Chapter 3 of **?**, which are applicable to relatively low-temperature biochemical reactions. The activity coefficients are calculated as a function of ionic strength (I), temperature, and charge of each species, without any other species-specific parameters. Using the default `Helgeson` method, the extended term parameter ("B-dot") is derived from data of **?** and Helgeson et al. (1981), and extrapolations of **?**.

Transformation of variables

Following the main workflow of CHNOSZ, `nonideal()` normally does not need to be used directly. Intead, invoke the calculations by setting the `IS` argument in `subcrt()` or `affinity()`. There are a few things to remember when using activity coefficients:

- H+ is assumed to behave ideally, so its activity coefficient is 1 for any ionic strength. You can calculate activity coefficients of H+ by setting `thermo$opt$ideal.H <<- FALSE`.
- Using `subcrt()` with `IS` not equal to zero, calculated values of `G` are the **adjusted** Gibbs energy at specified ionic strength (denoted by $\Delta G^\circ(I)$; **?**).
- Using `subcrt()` with `IS` not equal to zero, values in the `logact` argument stand for **log molality** of aqueous species in affinity calculations.
- Using `affinity()` with `IS` not equal to zero, the following values stand for **log molality** of aqueous species:
  - values of `logact` set by `basis()`;
  - values of `logact` set by `species()`.

- Using `equilibrate()` on the output of affinity calculated with `IS` not equal to zero, the following values stand for **log molality** of aqueous species:
  - the value of `loga.balance` used by `equilibrate()` (i.e., logarithm of total molality of the balancing basis species);
  - values of `loga.equil` returned by `equilibrate()`.

In other words, the activation of activity coefficients effects a transformation from activity to molality in the main workflow. A simple but comprehensive series of calculations demonstrating these tranformations is in `tests/testthat/test-logmolality.R`.

Because it is not possible to dynamically change the names of arguments in the functions, the user should be aware of the transformations mentioned above. However, the labels on diagrams *can* be automatically adjusted; accordingly, activities of aqueous species are relabeled as molalities by `diagram()` when `IS` is used in the calculation of `affinity()`.

Biochemical example

For the following calculations, we change the nonideality method to `Alberty`; this is a simpler formulation with parameters that are suitable for biochemical species at relatively low temperatures:

```
oldnon <- nonideal("Alberty")
```

Let's take a look at calculated activity coefficients at two temperatures and their effect on the standard Gibbs energies of formation ($\Delta G°f$) of species with different charge:

```
subcrt(c("MgATP-2", "MgHATP-", "MgH2ATP"),
       T = c(25, 100), IS = c(0, 0.25), property = "G")$out
```

```
## $`MgATP-2`
##     T       P       G    loggam   IS
## 1  25 1.00000 -773609  0.000000 0.00
## 2 100 1.01322 -780416 -0.656185 0.25
##
## $`MgHATP-`
##     T       P       G    loggam   IS
## 1  25 1.00000 -780994  0.000000 0.00
## 2 100 1.01322 -789617 -0.164046 0.25
##
## $MgH2ATP
##     T       P       G
```

```
## 1  25 1.00000 -786200
## 2 100 1.01322 -795012
```

The logarithms of the activity coefficients (`loggam`) are more negative for the higher-charged species, as well as at higher temperature, and have a stabilizing effect. That is, the adjusted Gibbs energies at $I > 0$ are less than the standard Gibbs energies at $I = 0$.

We can use these calculations to make some speciation plots, similar to Figures 1.2–1.5 in Alberty (2003). These figures show the distribution of differently charged species of adenosine triphosphate (ATP) as a function of pH, and the average number of H+ and Mg+2 bound to ATP in solution as a function of pH or pMg (-log$a$Mg+2).

Use `info()` to see what ATP species are available. The sources of high-temperature thermodynamic data for these species are two papers by LaRowe and Helgeson (**?**; **?**).

```
info(" ATP")
```

The plots for this system in Alberty's book were made for $I = 0.25$ M and $T = 25$ °C. As a demonstration of CHNOSZ's capabilities, we can assign a temperature of 100 °C.

```
T <- 100
```

Use the following commands to set the basis species, add the variously protonated ATP species, calculate the affinities of the formation reactions, equilibrate the system, and make a degree of formation ( ) or mole fraction diagram. This is similar to Figure 1.3 of Alberty (2003), but is calculated for $I = 0$ M and $T = 100$ °C:

To make the code more readable, commands for plotting titles and legends are not shown. All of the commands are available in the source of this document.

```
basis("MgCHNOPS+")
species(c("ATP-4", "HATP-3", "H2ATP-2", "H3ATP-", "H4ATP"))
a <- affinity(pH = c(3, 9), T = T)
e <- equilibrate(a)
d <- diagram(e, alpha = TRUE, tplot = FALSE)
```

Note that we have saved the numeric results of `diagram()`, i.e. the degrees of formation of the species ( ). With that, we can calculate and plot the average number of protons bound per ATP molecule. To do so, we use R's `rbind()` and `do.call()` to turn `alpha` into a matrix, then multiply by the number of protons bound to each species, and sum the columns to get the total (i.e. average proton number, NH+):

```
alphas <- do.call(rbind, d$plotvals)
nH <- alphas * 0:4
Hlab <- substitute(italic(N)[H^`+`])
plot(a$vals[[1]], colSums(nH), type = "l", xlab = "pH", ylab=Hlab, lty=2, col=2)
```

Adding the `IS` argument to `affinity()`, we can now plot NH+ at the given ionic strength. Here we set `plot.it = FALSE` in `diagram()` because we use the computed to make our own plot. This is similar to Figure 1.3 of Alberty (2003), but at higher temperature:

```
a <- affinity(pH = c(3, 9), IS = 0.25, T = T)
e <- equilibrate(a)
d <- diagram(e, alpha = TRUE, plot.it = FALSE)
alphas <- do.call(rbind, d$plotvals)
nH <- alphas * 0:4
lines(a$vals[[1]], colSums(nH))
```

Next, we add the Mg+2-complexed ATP species:

```
species(c("MgATP-2", "MgHATP-", "MgH2ATP", "Mg2ATP"))
```

Here is a function to calculate and plot NH+ for a given pMg:

```
Hplot <- function(pMg, IS = 0.25) {
  basis("Mg+2", -pMg)
  a <- affinity(pH = c(3, 9), IS = IS, T = T)
  e <- equilibrate(a)
  d <- diagram(e, alpha = TRUE, plot.it = FALSE)
  alphas <- do.call(rbind, d$plotvals)
  NH <- alphas * c(0:4, 0, 1, 2, 0)
  lines(a$vals[[1]], colSums(NH), lty = 7 - pMg, col = 7 - pMg)
}
```

With that function in hand, we plot the lines corresponding to pMg = 2 to 6. This is similar to Figure 1.4 of Alberty (2003):

```
plot(c(3, 9), c(0, 2), type = "n", xlab = "pH", ylab = Hlab)
lapply(2:6, Hplot)
```

The next function calculates and plots the average number of Mg+2 bound to ATP (NMg+2) for a given pH. Here we multiply `alpha` by the number of Mg+2 in each species, and negate log$a$Mg+2 (the variable used in `affinity()`) to get pMg:

```
Mgplot <- function(pH, IS = 0.25) {
  basis("pH", pH)
  a <- affinity(`Mg+2` = c(-2, -7), IS = IS, T = T)
  e <- equilibrate(a)
  d <- diagram(e, alpha = TRUE, plot.it = FALSE)
  alphas <- do.call(rbind, d$plotvals)
  NMg <- alphas * species()$`Mg+`
  lines(-a$vals[[1]], colSums(NMg), lty = 10 - pH, col = 10 - pH)
}
```

Using that function, we plot the lines corresponding to pH = 3 to 9. This is similar to Figure 1.5 of Alberty (2003):

```
Mglab <- substitute(italic(N)[Mg^`+2`])
plot(c(2, 7), c(0, 1.2), type = "n", xlab = "pMg", ylab = Mglab)
lapply(3:9, Mgplot)
```

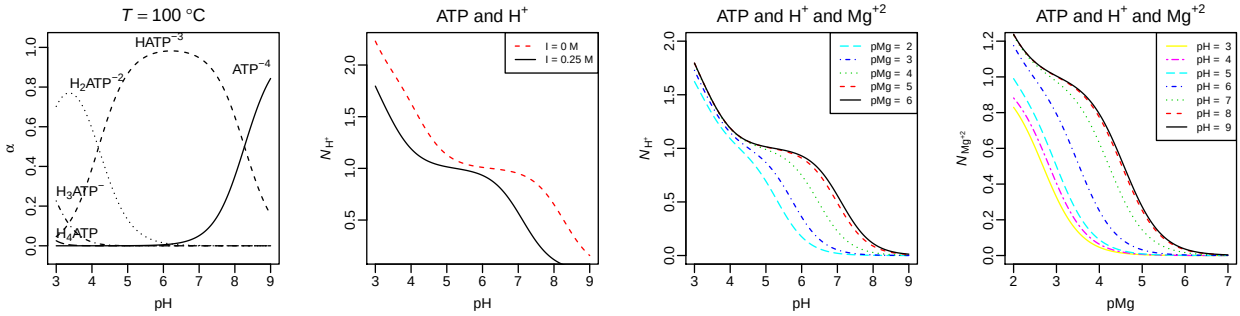


Figure 16: Binding of H⁺ and Mg⁺² to ATP at 100 °C and *I* = 0 M (first plot) or *I* = 0.25 M (third and fourth plots).

We have calculated the distribution of ATP species and average binding number of H+ and Mg+2 for given pH, pMg, ionic strength, and temperature. Accounting for the distribution of chemical species lends itself to thermodynamic models for reactions between reactants that have multiple ionized and complexed states. In contrast, Alberty (2003) and others propose models for biochemical reactions where the ionized and complexed species are combined into a single representation. Those models invoke Legendre-transformed thermodynamic properties, such as transformed Gibbs energies that are tabulated for specified pH, pMg, and ionic strength. Although the conceptual pathways are different, the two approaches lead to equivalent results concerning the energetics of the overall reactions and the conditions for equilibrium (**?**). The example here shows how the required calculations can be performed at the species level using conventional standard Gibbs energies for species referenced to infinite dilution (zero ionic strength). The effects of ionic strength are modeled "on the fly" in CHNOSZ by setting the `IS` argument in `subcrt()` or `affinity()` to invoke the nonideality model on top of the standard Gibbs energies of species.

Now that we're finished, we can reset the nonideality method to the default. (This really isn't needed here, because there aren't any nonideality calculations below):

```
nonideal(oldnon)
```

Proteins

Proteins in CHNOSZ are handled a little bit differently from other species. Amino acid group additivity is used to obtain the thermodynamic properties of proteins. Therefore, CHNOSZ has a data file with amino acid compositions of selected proteins, as well as functions for reading and downloading amino acid sequence data.

When proteins in CHNOSZ are identified by name, they include an underscore, such as in `LYSC_CHICK` (chicken lysozyme C). Use `pinfo()` to search for one or more proteins by name; multiple proteins from the same organism can be specified using the `organism` argument. The name search returns the rownumbers of `thermo$protein` (i.e. `iprotein`, the protein indices). Supply those protein indices to `pinfo()` to get the amino acid compositions:

```
p1 <- pinfo("LYSC_CHICK")
p2 <- pinfo(c("SHH", "OLIG2"), "HUMAN")
pinfo(c(p1, p2))
```

```
##     protein organism     ref    abbrv chains Ala Cys Asp Glu Phe Gly His Ile
## 6      LYSC    CHICK UniProt   P00698      1  12   8   7   2   3  12   1   6
## 452     SHH    HUMAN UniProt Q15465.N      1  14   3  10  14   5  16   6   7
## 453   OLIG2    HUMAN UniProt   Q13516      1  51   5  10  10   5  35  18   7
##     Lys Leu Met Asn Pro Gln Arg Ser Thr Val Trp Tyr
## 6     6   8   2  14   2   3  11  10   7   6   6   3
## 452  15  11   3   8   7   3  13  12   8   9   3   7
## 453  14  29  11   4  31   5  13  50  11  10   1   3
```

The length and chemical formula of one or more proteins are returned by `protein.length()` and `protein.formula()`. We can calculate the formula of the protein, and the per-residue formula, and show that both have the same average oxidation state of carbon:

These functions accept either names or the protein indices (`iprotein`).

```
pl <- protein.length("LYSC_CHICK")
pf <- protein.formula("LYSC_CHICK")
list(length = pl, protein = pf, residue = pf / pl,
     ZC_protein = ZC(pf), ZC_residue = ZC(pf / pl))
```

```
## $length
## [1] 129
##
## $protein
##              C   H   N   O  S
## LYSC_CHICK 613 959 193 185 10
##
## $residue
##                   C       H       N       O         S
## LYSC_CHICK 4.75194 7.43411 1.49612 1.43411 0.0775194
##
## $ZC_protein
## [1] 0.0163132
##
## $ZC_residue
## [1] 0.0163132
```

Group additivity and ionization

The group additivity calculations for proteins are based on equations and data from **?**, **?**, and **?**. There are two major options for the calculations: whether to calculate properties for crystalline or aqueous groups, and, for the latter, whether to model the ionization of the sidechain and terminal groups as a function of pH (and T and P). By default, additivity of aqueous groups is used, but the ionization calculations are not available in `subcrt()`:

```
subcrt("LYSC_CHICK")$out[[1]][1:6, ]
```

```
##        T       P      rho    logK        G         H       S       V      Cp
## 1   0.01 1.00000 0.999829 3217.68 -4021765 -10423733 3685.29 10049.2 4409.32
## 2  25.00 1.00000 0.997061 3019.80 -4119738 -10283083 4176.74 10421.0 6415.52
## 3  50.00 1.00000 0.988030 2861.40 -4230972 -10113250 4723.39 10600.2 7073.98
## 4  75.00 1.00000 0.974864 2734.31 -4355838  -9932209 5262.87 10708.1 7376.58
## 5 100.00 1.01322 0.958393 2632.00 -4493930  -9745475 5780.77 10782.9 7548.44
## 6 125.00 2.32014 0.939073 2549.29 -4644330  -9554920 6274.14 10840.9 7665.20
```

Let's compare experimental values of heat capacity of four proteins, from **?**, with those calculated using group additivity. After dividing Privalov and Makhatadze's experimental values by the lengths of the proteins to get per-residue values, we convert those to calories, then plot them.

```
PM90 <- read.csv(system.file("extdata/cpetc/PM90.csv", package = "CHNOSZ"))
plength <- protein.length(colnames(PM90)[2:5])
Cp_expt <- t(t(PM90[, 2:5]) / plength)
```

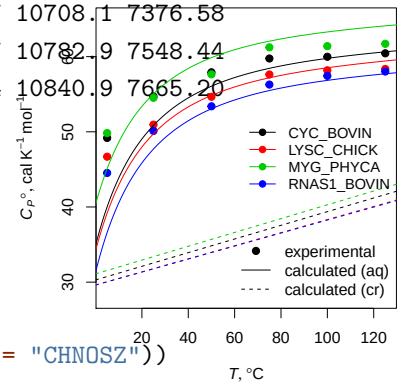


Figure 17: The heat capacity calculated by group additivity closely approximates experimental values for aqueous proteins. For a related figure showing the effects of ionization in the calculations, see ?ionize.aa.

```
matplot(PM90[, 1], convert(Cp_expt, "cal"), type = "p", pch = 19,
        xlab = axis.label("T"), ylab = axis.label("Cp0"), ylim = c(28, 65))
```

The loop calculates the properties of each protein using group additivity with aqueous or crystalline groups, then plots the per-residue values.

```
for(i in 1:4) {
  pname <- colnames(Cp_expt)[i]
  aq <- subcrt(pname, "aq", T = seq(0, 150))$out[[1]]
  cr <- subcrt(pname, "cr", T = seq(0, 150))$out[[1]]
  lines(aq$T, aq$Cp / plength[i], col = i)
  lines(cr$T, cr$Cp / plength[i], col = i, lty = 2)
}
legend("right", legend = colnames(Cp_expt),
       col = 1:4, pch = 19, lty = 1, bty = "n", cex = 0.9)
legend("bottomright", legend = c("experimental", "calculated (aq)",
       "calculated (cr)"), lty = c(NA, 1, 2), pch = c(19, NA, NA), bty = "n")
```

Although `subcrt()` has no provision for protein ionization, the properties of ionization can be calculated via `affinity()`, which calls `ionize.aa()` if a charged species is in the basis.

The following plot shows the calculated affinity of reaction between nonionized proteins and their ionized forms as a function of pH. Charged and uncharged sets of basis species are used to to activate and suppress the ionization calculations. The calculation of affinity for the ionized proteins returns multiple values (as a function of pH), but there is only one value of affinity returned for the nonionized proteins, so we need to use R's `as.numeric()` to avoid subtracting nonconformable arrays:

```
ip <- pinfo(c("CYC_BOVIN", "LYSC_CHICK", "MYG_PHYCA", "RNAS1_BOVIN"))
basis("CHNOS+")
a_ion <- affinity(pH = c(0, 14), iprotein = ip)
basis("CHNOS")
a_nonion <- affinity(iprotein = ip)
plot(c(0, 14), c(50, 300), xlab = "pH", ylab = axis.label("A"), type = "n")
for(i in 1:4) {
  A_ion <- as.numeric(a_ion$values[[i]])
  A_nonion <- as.numeric(a_nonion$values[[i]])
  lines(a_ion$vals[[1]], A_ion - A_nonion, col=i)
}
legend("topright", legend = a_ion$species$name,
       col = 1:4, lty = 1, bty = "n", cex = 0.9)
```

The affinity is always positive, showing the strong energetic drive for ionization of proteins in aqueous solution. The degrees of ion-

Whether to calculate properties using aqueous or crystalline groups is determined by the value of `thermo\$opt\$state`; if it is changed from its default of `aq` to `cr`, no ionization is possible.

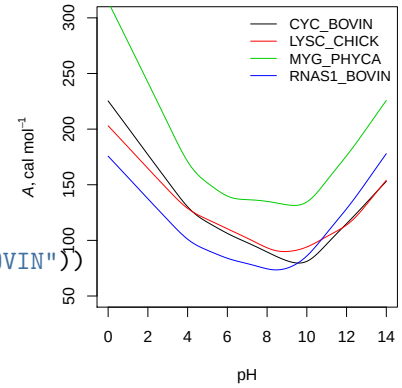


Figure 18: Affinity of ionization of proteins. See [demo(ionize)](../demo) for ionization properties calculated as a function of temperature and pH.