GCDModel – version 0.4.0

GCDmodel is a library to perform trace elements calculations for the composition of melts and minerals. The core functions share a common interface (parameters and return values of the same form), so they can easily be replaced and should probably be called by something like

do.call(MeltingFunctionName,args=list(kd= … *etc.*))

Two kind of functions are defined: melting functions and correction functions. Both accept arguments and return values of the same form. The only real difference is more one of convention: correction functions are meant to be ran after running a melting function, and as such correction functions expect to know about the trace element composition of the liquid. Melting functions, of course, have no such requirement.

Although all the functions accept all the arguments, they do not necessarily use all of them (the unused one default to something empty).

The following functions exist at the moment (and have very limited help available):

# Melting functions

BatchPM: Batch equilibrium, i.e. .

BatchPM(kd, c0, pm, cmins = matrix(), min.props, melt.arg = list(),

dont = character(0))

Used arguments:

kd is a *matrix* of partition coefficients, minerals in rows and elements in columns. Obviously the names of the minerals must match the names used in the mineral proportions. This function does not check the sanity of the Kd values supplied (in particular whether any value is missing; see, however, ppxCleanKdTable).

c0 is the composition of the source. The calculations will be done for elements that are both in Kd and in source.

pm (scalar) is the degree of partial melting (F), expressed in % so between 0 and 100

min.props is a named vector with mineral proportions in the solid. The names must be the same as used in kd. min.props is normalized internally by the function.

(cmins, melt.args, dont are accepted but not used by this function)

The function is somewhat tolerant to misformatted entry, and although it expects vectors for c0 and min.props, it *should* be able to deal gracefully with one-line matrices and even data.frames.

Return value : a **list** with the following elements:

* cL: named vector, composition of the liquid;
* cmins: matrix, composition of each mineral (found in min.props) for the calculated trace elements;
* cS: named vector, composition of the bulk solid;
* min.props: named vector, mineral proportions normalized to 1;
* kd: matrix, the partition coefficients actually used (i.e. subsetted for relevant mineral/elements);
* DD: named vector, the bulk distribution coefficient CS/CL.

Note that all the one-line information is returned as vectors and not as one-line matrices.

DummyPM : testing function.

This function is mostly provided for test purposes. Although it accepts the same arguments as BatchPM, and returns a list of the same form, cL contains only two variables, dummy1 and dummy2, with values of 1 and 2 respectively. The other elements are just passed from the input. This is useful for testing, to see if the external code is able to accommodate arbitrary variables.

# Correction functions

correctZrnSat and correctMnzSat are two very similar functions that correct the composition of the liquid taking into account zircon (respectively monazite) saturation. A third function, correctZrnMnzSat, is a wrapper that runs the previous 2 in succession.

correctZrnSat

Calculate zircon proportions based on Watson & Harrison (1983) saturation equation. First the value for Zr saturation in the melt is calculated and compared to the Zr contents of the melt (supplied as an input, i.e. one must run a “melting” function first). If the melt is undersaturated, nothing else happens. If it is oversaturated however, any Zr in excess of the saturation value is used to build zircon. The trace elements partitioning is then re-calculated taking into account the zircon (which affects, more or less, all trace elements). Zr amount in zircon is stoichiometric (497644 ppm). Zircon is noted “Zrn” (and this is the name under which its Kd should be entered in the Kd file).

Note that this function uses the saturation equation from GCDkit’s plugin “saturation” (function zrSaturation() ), and some ancillary GCDkit functions (millications() ).

The function accepts arguments similar to BatchPM. In addition to kd, c0, pm and min.props it does however require

cmins: matrix, the composition of minerals in trace elements (a such matrix is returned by BatchPM)

melt.arg: list containing the following elements:

* melt.arg$mjrs: named vector, major elements composition of the melt;
* melt.arg$trc: named vector, trace elements composition of the melt;
* melt.arg$TT: temperature in *Kelvin*.

dont: a character vector, containing the names of elements that should not be modified by this function (i.e. the return value for such elements will be taken from the input, cmins and melt.arg$trc). This is useful when chaining successive corrections.

Return value : a **list** with the following elements:

* cL: named vector, composition of the liquid;
* cmins: matrix, composition of each mineral (found in min.props) for the calculated trace elements;
* cS: named vector, composition of the bulk solid;
* min.props: named vector, mineral proportions normalized to 1, corrected (if relevant) to include zircon;
* kd: matrix, the partition coefficients taken from input (no subsetting is done in this function, for no other reason than « just because »).
* DD: named vector, the bulk distribution coefficient CS/CL.
* dont = “Zr”, for further treatment by subsequent correction functions.

correctMnzSat

Calculate monazite saturation based on Montel et al. (1992) equations taking into account the partitioning of individual LREE, and modified by Montel (1996) to treat Th in the same way. The results will thus be slightly different from Montel et al. (1992)’s “all REE” saturation. The logic of the calculation is similar to zircon (i.e. excess LREE over saturation are assigned to monazite).

The arguments and return values of this function are very similar to the previous. Elements modified (dont in the return value) are *Th, La, Ce, Pr, Nd, Sm, Gd*. Monazite proportions is noted as “Mnz” (and this is the name under which its Kd should be entered in the Kd file).

The saturation is calculated using GCDkit’s millications() and mzSaturationWithTh().

The water amount in the melt, melt.arg$H2O, must also be supplied.

correctZrnMnzSat

Chain the previous two functions. It takes the same arguments and returns the same values. Zircon is calculated first, and monazite second; monazite correction is instructed (via “dont" to not further modify Zr concentrations, so Zr in monazite will not be calculated.

Further melting/correction functions may be developed; they will use the same interface (in/out) so they should be interchangeable.

# Other functions

ppxCleanKdTable

A convenience function that cleans a Kd table in doing the following:

1. Make sure the Kd table is a matrix (not a data.frame)
2. Add a row of zeros for any mineral supplied in the arguments and not present in the Kd file;
3. If a mineral comes in two varieties in the arguments (i.e. Mica\_1 and Mica\_2), and a matching row (i.e. Mica) is found in Kd file, duplicate this row and add the suffixes;
4. Replace all NA’s by zeros.

So for instance if a Kd table contains entries for q, Kfs and Pl, and the mineral list ppxPhases is q, Kfs, Mica the return table includes q, Kfs, Pl and Mica (the latter with only zeros). If ppxPhases contains q, Pl\_1 and Pl\_2 the return table contains q, Kfs, Pl, Pl\_1 and Pl\_2 (the latter three being exactly similar). Use this in case of solvus…

ppxCleanKdTable (kd,ppxPhases,interactive=T)

kd: a kd table with elements names in columns and minerals in rows.

ppxPhases: the list of phases that should be present in the cleaned table

interactive: whether to give a pop up message or not when rows are added.

Returns : a matrix of Kd.

# Example

library(GCDkit)

library(GCDmodel)

data(sazava)

accessVar("sazava")

trc<-c("Rb","Sr","Ba","Zr")

kd<-matrix(c(0.2,2,0.8,0.01,

1.5,0.5,2,0.01,

2,0.9,0.2,0.2,

0.01,0.01,0.01,0),

nrow=4,byrow=F)

colnames(kd)<-trc

rownames(kd)<-c("Pl","Kfs","Bt","Zrn")

c0<-WR["Sa-1",trc]

mp<-c(50,30,20)

names(mp)<-c("Pl","Kfs","Bt")

batch<-BatchPM(kd=kd,c0=c0,pm=25,min.props=mp)

# No zircon !

my.arg<-list(mjrs=WR["Sa-1",major],trc=batch$cL,TT=1073)

zr.corr<-correctZrnSat(kd=kd,c0=c0,pm=25,min.props=mp,cmins=batch$cmins,melt.arg=my.arg)

zr.corr

# Zircon !

my.arg2<-list(mjrs=WR["Sa-1",major],trc=batch$cL,TT=973)

zr.corr2<-correctZrnSat(kd=kd,c0=c0,pm=25,min.props=mp,cmins=batch$cmins,melt.arg=my.arg2)

zr.corr2