Package 'GCDkit.Mineral'

May 23, 2023

Version 1.0.0

Date 2023-05-25

Title Mineralogical Data Toolkit for Windows

Author Vojtech Janousek [aut, cre],
Vojtech Erban [aut],
Colin Farrow [aut]

Maintainer Vojtech Janousek <vojtech.janousek@geology.cz>

Depends R (>= 3.6),
methods,
tcltk

Imports compiler,
foreign,
graphics,
grDevices,
grid,
IRdisplay,
lattice,
MASS,
R2HTML,
sp,
stats,
utils

Suggests RODBC,
XML

Description A program for recalculation and plotting of mineral analyses.
With complete graphical user interface (GUI) it runs under Windows 7/8/10/11,
complete functionality/stability under earlier Windows versions cannot be guaranteed.
Should also work on Mac OS X (release 10.6 and above) and various distributions of
Linux (Debian, RedHat, SUSE, Ubuntu).

License GPL (>= 2)

URL http://www.gcdkit.org/mineral

LazyData false

R topics documented:

.callGCDkit 4

about . . . 5
accessVar . . . 6
Add contours . . . 7
alumosilicates . . . 8
amphibole . . . 9
apatite . . . 9
assign1col . . . 10
assign1symb . . . 11
assignColLab . . . 11
assignColVar . . . 13
assignSymbGroup . . . 14
assignSymbLab . . . 15
assignSymbLett . . . 16
atacazo . . . 17
binary . . . 18
binaryBoxplot . . . 20
blatna . . . 21
Boolean conditions . . . 22
bpplot2 . . . 23
calc . . . 25
calcCore . . . 26
clr.transform . . . 27
cluster . . . 30
combined . . . 31
contourAll . . . 32
contourGroups . . . 34
crosstab . . . 36
cutMy . . . 37
Edit labels . . . 38
Edit numeric data . . . 38
editLabFactor . . . 39
Export to Access . . . 40
Export to DBF . . . 41
Export to Excel . . . 42
Export to HTML tables . . . 43
feldspars . . . 46
figAdd . . . 46
figAddReservoirs . . . 49
figaro.identify . . . 51
figCol . . . 52
figEdit . . . 53
figGbo . . . 54
figMulti . . . 54
figOverplot . . . 56
figRedraw . . . 58
figScale . . . 59
figUser . . . 60
figZoom . . . 61
filledContourFig . . . 63
formula2vector . . . 65
garnet . . . 66
gcdOptions . . . 66

graphicsOff . . . 69
groupsByLabel . . . 70
HTMLFormula . . . 70
ID . . . 72
info . . . 73
joinGroups . . . 73
loadData . . . 74
mergeData . . . 78
micas . . . 79
minAllocateAtoms . . . 79
minAssign . . . 81
minCheckValency . . . 83
minClassify . . . 84
minComp . . . 85
minDat . . . 87
minEndMembers . . . 87
mineral-class . . . 89
mineral.db-class . . . 92
minFormula . . . 93
minMain . . . 96
minValues . . . 98
Molecular weights . . . 99
Multiple plots . . . 100
olivine . . . 103
oxide2oxide . . . 104
oxide2ppm . . . 105
pairsCorr . . . 105
pdfAll . . . 108
peekDataset . . . 109
peterplot . . . 110
phasePropPlot . . . 112
Plate . . . 114
Plate editing . . . 116
plateLabelSlots . . . 118
plotWithCircles . . . 120
pokeDataset . . . 122
ppm2oxide . . . 123
prComp . . . 124
printSamples . . . 125
printSingle . . . 127
profiler . . . 128
psAll . . . 130
purgeDatasets . . . 131
pyroxene . . . 132
quitGCDkit . . . 133
r2clipboard . . . 133
recalcOptions . . . 134
recast . . . 135
Regular expressions . . . 136
sampleDataset . . . 137
saveData . . . 138
saveResults . . . 139

sazava . . . 139
scattersmooth . . . 140
selectAll . . . 143
selectByLabel . . . 144
selectByMineral . . . 145
selectColumnLabel . . . 146
selectColumnsLabels . . . 147
selectPalette . . . 149
selectSubset . . . 151
setCex . . . 153
setShutUp . . . 154
setTransparency . . . 155
showColours . . . 156
showLegend . . . 157
showSymbols . . . 159
statsByGroup . . . 159
statsByGroupPlot . . . 160
strip . . . 161
stripBoxplot . . . 162
Subset by range . . . 164
summaryAll . . . 165
summaryByGroup . . . 166
summarySingle . . . 168
summarySingleByGroup . . . 170
ternary . . . 171
threeD . . . 173
tkSelectVariable . . . 175
tk_winDialog . . . 176
tk_winDialogString . . . 177
trendTicks . . . 178
webMineral . . . 181
wholeRock . . . 182

Index **184**

`.callGCDkit` *Plotting recalculated mineral analyses*

Description

This is an internal function that allows interfacing to the standard *GCDkit* functions with recalculated data for a single mineral (apfu, atoms allocated to crystallographic positions, extra values, end-member proportions).

Usage

```
.callGCDkit(fun,mineral=NULL,what=NULL)
```

Arguments

`fun`	character; function with all arguments to be executed
`mineral`	character; mineral name
`what`	character; slot name

Details

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

binary, ternary, threeD, threeD

plotWithCircles, binaryBoxplot, multiple

peterplot, profiler, figaro,

Plate, Plate editing, plateUser

Examples

```
data(combined)
accessVar("combined")
groupsByLabel("Mineral")

# Binary plot - apfu for clinopyroxene
.callGCDkit("binary('Si','Al')",mineral="clinopyroxene",what="recalc")

# Binary plot - Si and Al in clinopyroxene tetragonal site
.callGCDkit("binary('Si_T','Al_T')",mineral="clinopyroxene",what="formula")

# Ternary plot - apfu for feldspar
.callGCDkit("ternary('K','Ca','Na')",mineral="feldspar",what="recalc")

# Multiple plots - apfu for feldspar
.callGCDkit("multiple('Si',c('Al','Ca','Na'))",mineral="feldspar",what="recalc")
plateUser(las=3,cex=2.5,col="darkred",cex.axis=1.5,cex.lab=1.8)

# Plot with circles - apfu for feldspar
.callGCDkit("plotWithCircles('Al','Ca','Na')",mineral="feldspar",what="recalc")

# binaryBoxplot - Si and Al in clinopyroxene tetragonal site
.callGCDkit("binaryBoxplot('Si_T','Al_T')",mineral="clinopyroxene",what="formula")

# Anomaly plot - apfu for feldspar
.callGCDkit("peterplot('Al','Ca','Na')",mineral="feldspar",what="recalc")

# Profiler - apfu for feldspar
.callGCDkit("profiler('Si',c('Al','Ca','Na'),pch=1:3,
    col=1:3,col.lines=1:3)",mineral="feldspar",what="recalc")
```

---

about *About GCDkit*

Description

Prints short information about the current version of GCDkit and contact addresses of its authors.

Usage

```
about()
```

Arguments

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

---

accessVar *Accessing data in memory of R*

Description

Loads data already present in memory of R into GCDkit.

Usage

```
accessVar(var = NULL, source.first = TRUE, source.diagrams = TRUE, source.plugins = TRUE,
    poke.data = TRUE, GUI = FALSE)
```

Arguments

var	numeric matrix, or a text string specifying a name of such a variable to be accessed
source.first	logical; should be the GCDkit.r sourced?
source.diagrams	logical; should be built the lists of available diagrams?
source.plugins	logical; should be all the plugins sourced?
poke.data	logical; should the dataset be stored in the WRCube?
GUI	logical; is the function called from GUI (or from the command line)?

Details

This function enables accessing a variable already present in R. The variable is a numeric matrix or dataframe that can be either passed directly, or referred to by its name. Typically is the function accessVar used to access sample data sets, previously made available using the command data. See Examples.

If source.diagrams = TRUE, then two lists of applicable diagrams are built (based on data present in the current dataset), claslist and tectlist.

Value

WR	numeric matrix: all numeric data
labels	data frame: all at least partly character fields; labels$Symbol contains plotting symbols and labels$Colour the plotting colours

The function prints a short summary about the attached data. It also loads and executes the Plugins, i.e. all the R code that is currently stored in the subdirectory '\Plugin'.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(swiss)
accessVar(swiss)
binary("Catholic","Education")

data(sazava)
accessVar("sazava")
binary("SiO2","Ba")
```

`Add contours` *Add contours*

Description

Superposes contour lines to a Figaro-compatible plot.

Usage

```
addContours(GUI = FALSE, bandwidth = "auto",...)
```

Arguments

`GUI`	logical; is the function called from GUI (or in a direct mode)?
`bandwidth`	vector of bandwidths for x and y directions provided to the function `kde2d`. See Details.
`...`	additional parameters passed to the underlying function `contour`. Typically plotting parameters.

Details

This is, in principle, a front end to the standard R function `contour`. It will work on both the stand-alone Figaro-compatible plot or a plate thereof.

The `bandwidth` should be a positive number or 'auto', whereby the higher value corresponds to a smoother result. The necessary calculations are done by the function `kde2d`.

Value

None.

Author(s)

Vojtěch Erban, <erban@sopky.cz> Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'`filled.contour`' '`kde2d`' '`par`' '`figaro`'

Examples

```
data(atacazo)
accessVar("atacazo")

# single plot
binary("SiO2","MgO")
addContours(col="darkblue",lty="dashed",bandwidth=1.5)

windows()
figRedraw()
addContours(col="darkgreen",lty="dotted",bandwidth=5)

# multiple plot
multiple("SiO2","Al2O3,MgO,CaO,K2O")
plateCex(2)
plateCexLab(1.5)
addContours(col="darkgreen",lty="dashed")
```

`alumosilicates` *Selected analyses of alumosilicates*

Description

This data set gives the selected chemical analyses of alumosilicates from *Deer et al. (2013)*.

Usage

```
data(alumosilicates)
```

Format

A data frame containing 5 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 37 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(alumosilicates)
accessVar("alumosilicates")
print(WR)
```

amphibole *Selected amphibole analyses*

Description

This data set gives the selected chemical analyses of amphibole from *Deer et al. (2013)*.

Usage

```
data(amphibole)
```

Format

A data frame containing 14 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 140 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(amphibole)
accessVar("amphibole")
print(WR)
```

apatite *Selected apatite analyses*

Description

This data set gives the selected chemical analyses of apatite from *Deer et al. (2013)*.

Usage

```
data(apatite)
```

Format

A data frame containing 5 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 475 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(apatite)
accessVar("apatite")
print(WR)
```

---

assign1col *Uniform colours*

Description

Assigns the same plotting colour to all samples.

Usage

```
assign1col(col=-1)
```

Arguments

col	numeric; code of the colour.

Details

This function sets the same colour to all of the plotting symbols. If 'col' = -1 (the default), the user is prompted to specify its code.

Value

Sets 'labels$Colour' to code of the selected plotting colour.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

To display the current legend use showLegend. Symbols and colours by a single label can be assigned by assignSymbLab and assignColLab respectively, symbols and colours by groups simultaneously by assignSymbGroup. Uniform symbols are obtained by assign1symb. Table of available plotting symbols is displayed by showSymbols and colours by showColours.

assign1symb *Uniform symbols*

Description

Assigns the same plotting symbol to all samples.

Usage

```
assign1symb(pch=-1)
```

Arguments

pch	numeric; code of the plotting symbol.

Details

This function sets the same plotting symbol to all the data points. If 'pch' = -1 (the default), the user is prompted to specify its code.

Value

Sets 'labels$Symbol' to code of the selected plotting symbol.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

To display the current legend use showLegend. Symbols and colours by a single label can be assigned by assignSymbLab and assignColLab respectively, symbols and colours by groups simultaneously by assignSymbGroup. Uniform colours are obtained by assign1col. Table of available plotting symbols is displayed by showSymbols and colours by showColours.

assignColLab *Colours by label*

Description

Assigns plotting colours according to the levels of the chosen label or, alternatively, sample names.

Usage

```
assignColLab(lab = NULL, pal = NULL, colours = NULL, display.legend = FALSE)
```

Arguments

lab	specification of the variable to be used for colours assignment. See Details.
pal	character; name of the palette to be used when no colours are specified directly. Batch mode only.
colours	a vector with codes of colours to be assigned. Batch mode only.
display.legend	logical; should be the legend displayed? Batch mode only.

Details

If called from in interactive mode (from GUI), the variable (sample names or label) can be selected using the function 'selectColumnLabel'.

In batch mode, 'lab' can be an integer (1 for sample names, or a sequence number of the column in the 'labels' plus 1). Alternatively, it can contain the full name of a column in 'labels'. See examples.

If in batch mode, either 'colours' or 'palette' have to be specified for the correct colour assignment.

Value

Sets 'leg.col' to a sequence number of column in 'labels' that is to be used to build the legend or -1 if sample numbers are to be used; 'labels$Colour' contains the codes of the desired plotting colours.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

To display the current legend use showLegend. Symbols by a single label can be assigned by assignSymbLab, symbols and colours by groups simultaneously by assignSymbGroup. Uniform colours and symbols are obtained by assign1symb and assign1col. Table of available plotting symbols is displayed by showSymbols and colours by showColours.

Selecting a label: selectColumnLabel.

Selecting a palette: selectPalette.

Examples

```
data(sazava)
accessVar("sazava")

## Not run: assignColLab()    # Interactive mode

# Sample names, standard GCDkit colours palette
assignColLab(1,colours=palette.gcdkit,display.legend=TRUE)

# Standard palettes
assignColLab(3,pal="jet.colours",display.legend=TRUE)        # Second column in labels

assignColLab("Locality",pal="jet.colours",display.legend=TRUE) # Ditto (here Locality)

# User defined palette
```

```
my.palette<-colorRampPalette(c("black", "green", "red"),space = "rgb")
assignColLab("Locality",pal="my.palette",display.legend=TRUE)
```

---

assignColVar *Colours by a variable*

Description

Assigns plotting colours according to the values of the variable.

Usage

```
assignColVar(what=NULL,pal="heat.colours",save=TRUE,n=15,range=NULL,
    quant=0,eq.classes=FALSE,alt.leg=FALSE)
```

Arguments

what	variable name or a formula; if NULL a dialogue is displayed
pal	character; name of a palette
save	logical;should the newly picked colours be assigned to 'labels'?
n	desired approximate number of colours to be assigned.
range	numeric vector with two items; (optional) desired range of the variable to be covered.
quant	numeric, 0-50; quantile to be potentially used to get rid of outliers. See details.
eq.classes	logical; should classes contain equal number of values?
alt.leg	logical; should be the alternative (continuous) legend shown? See Examples.

Details

For selection of the variable is employed the function 'selectColumnLabel'. The user can specify either existing data column in the 'WR' or a formula. The colours can be optionally (default behaviour) assigned globally, so that all the plots will use these from this point on. If not specified upon function call, the palette is picked using selectPalette. The possible values are: 'grays','reds','blues','greens','cyans','violets','yellows','cm.colors', 'heat.colors','terrain.colors','topo.colors','rainbow' and 'jet.colors'.

Also, user-defined palette functions are supported. See Examples.

The analyses with no data available for the colours assignment will remain black.

If quant differs from the default value of zero, the data are trimmed to an interval (quant,100-quant)-th quantile of the dataset and all values out of it plotted in gray.

Setting eq.classes=TRUE allows to have classes with equal number of values (as opposed to equal intervals). This option is best suited for very skewed datasets (lots of points with similar values, some outliers).

Value

A list of two components, col and leg. The former are the plotting colours, the latter contains information needed to build a legend. If save = TRUE, 'labels$Colour' will acquire the codes of desired plotting colours.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Jean-François Moyen, <jfmoyen@gmail.com>

See Also

quantile Colours by a single variable can be assigned by assignColLab, symbols and colours by groups simultaneously by assignSymbGroup. Uniform colours are obtained by assign1col. Table of available plotting colours is obtained by showColours.

Examples

```
data(atacazo)
accessVar("atacazo")

assignColVar("Na2O/K2O","greens")
binary("SiO2","MgO")

showLegend()
showLegend(alt.leg=TRUE)

my.palette<-colorRampPalette(c("black", "darkgreen", "red"),space = "rgb")
assignColVar("SiO2","my.palette")
binary("SiO2","MgO")
figLegend(x="topright",bg="#FFFFFFAA",alt.leg=TRUE,just.colours=TRUE)

assignColVar("SiO2","my.palette",n=7,quant=5)
binary("SiO2","MgO")
figLegend(x="topright",bg="#FFFFFFAA",alt.leg=TRUE,just.colours=TRUE)
```

---

assignSymbGroup *Symbols/colours by groups*

Description

Lets the user to assign plotting symbols and colours according to the levels of the defined groups.

Usage

```
assignSymbGroup()
```

Arguments

None.

Value

Sets 'leg.col' and 'leg.pch' to zero, 'labels$Symbol' contains the codes of desired plotting symbols, 'labels$Colour' of plotting colours.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

To display the current legend use showLegend. Symbols by a single label can be assigned by assignSymbLab, colours using assignColLab. Uniform colours and symbols are obtained by assign1symb and assign1col. Table of available plotting symbols is displayed by showSymbols and colours by showColours.

---

assignSymbLab *Symbols by label*

Description

Assigns plotting symbols according to the levels of the chosen label or, alternatively, sample names.

Usage

```
assignSymbLab(lab = NULL, symbols = NULL, display.legend = FALSE)
```

Arguments

lab	specification of the variable to be used for symbols assignment. See Details.
symbols	a vector with codes of plotting symbols to be assigned. Batch mode only.
display.legend	logical; should be the legend displayed? Batch mode only.

Details

If called from in interactive mode (from GUI), the variable (sample names or label) can be selected using the function 'selectColumnLabel'.

In batch mode, 'lab' can be an integer (1 for sample names, or a sequence number of the column in the 'labels' plus 1). Alternatively, it can contain the full name of a column in 'labels'. See examples.

If in batch mode, 'symbols' have to be specified for the correct plotting symbols assignment.

Value

Sets 'leg.pch' to a sequence number of column in 'labels' that is to be used to build the legend or -1 if sample numbers are to be used; 'labels$Symbol' contains the codes for desired plotting symbols.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

To display the current legend use showLegend.

Using the function assignSymbLett, initial letters of the respective levels of the chosen label can be assigned to the plotting symbols.

Colours by a single label can be assigned by assignColLab, symbols and colours by groups simultaneously by assignSymbGroup. Uniform colours and symbols are obtained by assign1symb and assign1col. Table of available plotting symbols is displayed by showSymbols and colours by showColours.

Selecting a label: selectColumnLabel.

Examples

```
data(sazava)
accessVar("sazava")

## Not run: assignSymbLab()              # Interactive mode

# Sample names, standard GCDkit colours palette
assignSymbLab(1,symbols=1:nrow(WR),display.legend=TRUE)            # By samples

assignSymbLab(2,symbols=c("+","*","@"),display.legend=TRUE)      # First column in labels

assignSymbLab("Intrusion",symbols=c(12,15,17),display.legend=TRUE) # Ditto (here Intrusion)
```

`assignSymbLett` *Symbols by label - initial letters*

Description

Assigns plotting symbols to initial letters of the respective levels of the chosen label.

Usage

```
assignSymbLett(lab = NULL, display.legend = FALSE)
```

Arguments

`lab`	specification of the variable to be used for symbols assignment. See Details.
`display.legend`	logical; should be the legend displayed? Batch mode only.

Details

If called from in interactive mode (from GUI), the variable (sample names or label) can be selected using the function '`selectColumnLabel`'.

In batch mode, '`lab`' can be an integer (a sequence number of the column in the '`labels`'). Alternatively, it can contain the full name of a column in '`labels`'. See examples.

Value

Sets '`leg.pch`' to a sequence number of column in '`labels`' that is to be used to build the legend; '`labels$Symbol`' contains the plotting symbols, which correspond to initial letters for the levels of the specified label.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

To display the current legend use `showLegend`. Symbols by a single label can be assigned by `assignSymbLab`, colours by `assignColLab`, symbols and colours by groups simultaneously by `assignSymbGroup`. Uniform colours or symbols are achieved by `assign1symb` and `assign1col`. Table of available plotting symbols is displayed by `showSymbols` and colours by `showColours`.

Examples

```
data(sazava)
accessVar("sazava")

## Not run: assignSymbLett()                      # Interactive mode

assignSymbLett(2,display.legend=TRUE)           # Second column in labels
assignSymbLett("Locality",display.legend=TRUE)  # The same (here Locality)
```

`atacazo` *Whole-rock composition of lavas from the Atacazo and Ninahuilca volcanoes, Ecuador*

Description

This data set gives the whole-rock major- and trace-element contents, together with Sr and Nd isotopic compositions of lavas from two volcanic complexes in Ecuador: the Atacazo and the Ninahuilca *(Hidalgo, 2006; Hidalgo et al., 2008)*. This dataset is used in a worked example (chapter 25) of *Janousek et al.*'s book (2016).

Note that this data set contains information on symbols and colours to be used in *GCDkit*, as well as labels (`Volcano`) that can be used for grouping or similar purposes. It also includes `87Sr/86Sr` and `143Nd/144Nd`. Therefore, if the SrNd plugin for *GCDkit* is installed, these columns will automatically be recognized as Sr and Nd initial isotopic ratios when loading it into *GCDkit* (via `accessVar("atacazo")`, allowing variables such as TDM to be calculated and isotope-based diagrams to be plotted. As no `Age` column is supplied, the user will be prompted for the emplacement age; the volcanoes being Quaternary in age (220-71 ka for Atacazo and 71-2 ka for Ninahuilca), the age correction is insignificant and a small value (of 0.1 for instance) is adequate.

Usage

```
data(atacazo)
```

Format

A data frame containing 110 observations of 38 variables.

Source

data by Silvana Hidalgo, <shidalgo@igepn.edu.ec>,

formatted by Jean-François Moyen, <jfmoyen@gmail.com>

References

Hidalgo S (2006) Les interactions entre magmas calco-alcalins "classiques" et adakitiques: exemple du complexe volcanique Atacazo-Ninahuilca (Equateur). Unpublished PhD thesis, Universite Blaise-Pascal, Clermont-Ferrand, France

Hidalgo S, Monzier M, Almeida E, Chazot G, Eissen JP, van der Plicht J, Hall M (2008) Late Pleistocene and Holocene activity of the Atacazo-Ninahuilca Volcanic Complex (Ecuador). J Volc Geoth Res 176: 16-26 doi: 10.1016/j.jvolgeores.2008.05.017

Janousek V, Moyen JF, Martin H, Erban V, Farrow CM (2016) Geochemical Modelling of Igneous Processes - Principles and Recipes in the R Language. Springer Verlag, Berlin doi: 10.1007/9783-662467923

Examples

```
data(atacazo)
accessVar("atacazo")

binary("SiO2","Ba")

## Not run: ageEps()    # Works only in GCDkit proper
```

`binary` *Binary plot*

Description

These functions display data as a binary plot.

Usage

```
binary(x=NULL,y=NULL,log="",samples=rownames(WR),
    new=TRUE, ...)

plotWithLimits(x.data, y.data,
    digits.x=NULL, digits.y=NULL,log = "",new = TRUE,
    xmin=.round.min.down(x.data,dec.places=digits.x,expand=TRUE),
    xmax=.round.max.up(x.data,dec.places=digits.x,expand=TRUE),
    ymin=.round.min.down(y.data,dec.places=digits.y,expand=TRUE),
    ymax=.round.max.up(y.data,dec.places=digits.y,expand=TRUE),
    xlab = "", ylab = "", fousy = "",
    IDlabels=getOption("gcd.ident"), fit = FALSE, main = "",
    pch = labels[names(x.data), "Symbol"],
    col = labels[names(x.data), "Colour"],
   cex=labels[names(x.data),"Size"],title=NULL,xaxs="i",yaxs="i",interactive=FALSE)
```

Arguments

`x,y`	character; specification of the plotting variables (formulae OK).
`log`	a vector `''`,`'x'`,`'y'` or `'xy'` specifying which of the axes are to be logarithmic
`samples`	character or numeric vector; specification of the samples to be plotted.
`new`	logical; should be opened a new plotting window?
`...`	Further parameters to the function '`plotWithLimits`'.
`x.data`	a numerical vector with the x data.
`y.data`	a numerical vector with the y data.
`digits.x`	Precision to which should be rounded the x axis labels.
`digits.y`	Precision to which should be rounded the y axis labels.
`xmin, xmax`	limits of the x axis.
`ymin, ymax`	limits of the y axis.
`xlab, ylab`	labels for the x and y axes, respectively.

fousy	numeric vector: if specified, vertical error bars are plotted at each data point.
IDlabels	labels that are to be used to identify the individual data points
fit	logical, should be the data fitted by a least squares line?
main	main title for the plot.
pch	plotting symbols.
col	plotting colours.
cex	relative size of the plotting symbols.
title	title for the plotting window.
xaxs, yaxs	type of the x and y axes.
interactive	logical; for internal use by our French colleagues.

Details

The function 'plots.with.limits' sets up the axes, labels them, plots the data and, if desired, enables the user to identify the data points interactively.

'binary' is the user interface to 'plotWithLimits'.

The variables to be plotted are selected using the function 'selectColumnLabel'. In the specification of the variables can be used also arithmetic expressions, see calcCore for the correct syntax.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSubset for details.

The functions are Figaro-compatible.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

plot

Examples

```
data(sazava)
accessVar("sazava")

binary("K2O/Na2O","Rb")

binary("Rb/Sr","Ba/Rb",log="xy",samples=1:10,col="red",pch="+",main="My plot")

plotWithLimits(WR[,"SiO2"]/10,WR[,"Na2O"]+WR[,"K2O"],xlab="SiO2/10",
    ylab="alkalis")

plotWithLimits(WR[,"Rb"],WR[,"Sr"],xlab="Rb",ylab="Sr",log="xy")

plotWithLimits(WR[,"SiO2"],WR[,"Ba"],fousy=WR[,"Ba"]*0.05,fit=TRUE)
```

`binaryBoxplot` *Binary boxplot*

Description

A binary plot combined with boxplots for both variables.

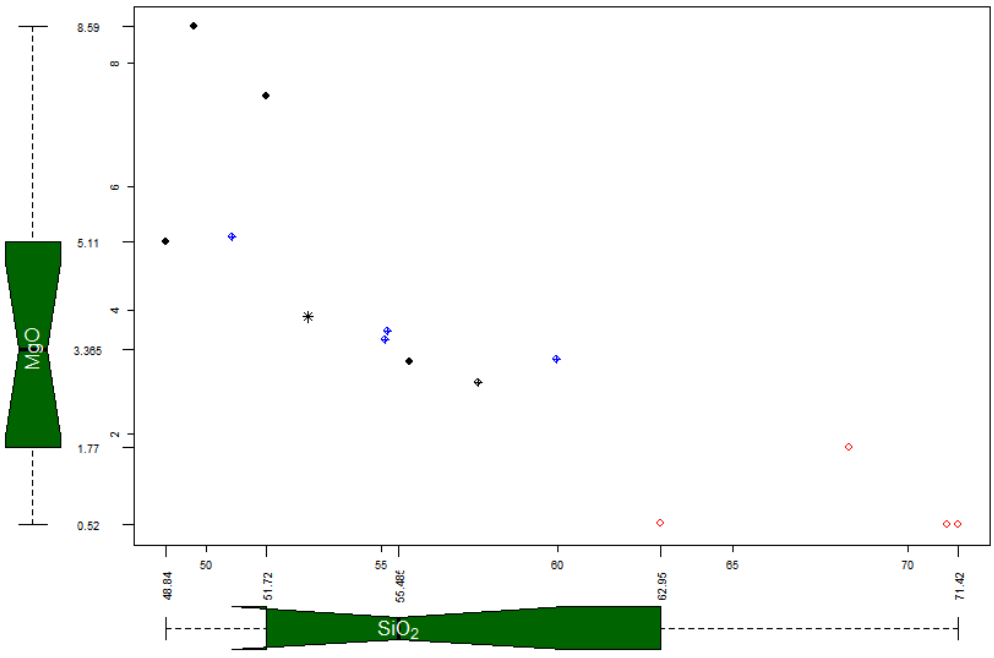
Usage

```
binaryBoxplot(xaxis="",yaxis="")
```

Arguments

`xaxis, yaxis`	specification of the variables. Formulae are OK.

Details



Unless specified in the call, the variables to be plotted are selected using the function '`selectColumnLabel`.

In the specification of the variables can be used also arithmetic expressions, see `calcCore` for the correct syntax.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSubset` for details.

Value

None.

Warning

This function IS NOT Figaro-compatible.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

plot boxplot

Examples

```
data(sazava)
accessVar("sazava")

binaryBoxplot("SiO2/10","Na2O+K2O")
```

---

blatna — *Whole-rock geochemical composition of the Blatná suite, Central Bohemian Plutonic Complex*

Description

These data sets give the whole-rock major- and trace-element contents as well as Sr-Nd isotopic composition in selected samples (monzogabbros, hybrid quartz monzonites and granodiorites) of the c. 345 Ma old high-K calc-alkaline Blatná suite of the Variscan Central Bohemian Plutonic Complex (Bohemian Massif, Czech Republic).

Usage

```
data(blatna)
data(blatna_iso)
```

Format

A data frame containing 11 (`blatna`) and 9 (`blatna_iso`) observations, respectively.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Janoušek V, Rogers G, Bowes DR (1995) Sr-Nd isotopic constraints on the petrogenesis of the Central Bohemian Pluton, Czech Republic. Geol Rundsch 84: 520-534 doi: 10.1007/BF00284518

Janoušek V, Bowes DR, Rogers G, Farrow CM, Jelínek E (2000) Modelling diverse processes in the petrogenesis of a composite batholith: the Central Bohemian Pluton, Central European Hercynides. J Petrol 41: 511-543 doi: 10.1093/petrology/41.4.511

Janoušek V, Wiegand B, Žák J (2010) Dating the onset of Variscan crustal exhumation in the core of the Bohemian Massif: new U-Pb single zircon ages from the high-K calc-alkaline granodiorites

of the Blatná suite, Central Bohemian Plutonic Complex. J Geol Soc (London) 167: 347-360 doi: 10.1144/001676492009008

Janoušek V, Erban Kochergina YV, Andronikov A, Kusbach V (2022) Decoupling of Mg from Sr-Nd isotopic compositions in Variscan subduction-related plutonic rocks from the Bohemian Massif: implications for mantle enrichment processes and genesis of orogenic ultrapotassic magmatic rocks. Int J Earth Sci 111: 1491-1518. doi: 10.1144/001676492009008

See Also

data, accessVar

Examples

```
data(blatna)
accessVar("blatna")
binary("SiO2","Ba")

## Not run:  # Works only in GCDkit proper
    data(blatna_iso)
    accessVar("blatna_iso")
    ageEps()

## End(Not run)
```

Boolean conditions *Select subset by Boolean condition*

Description

Selecting subsets of the current dataset using Boolean conditions that can query both numeric fields and labels. Regular expressions can be employed to search the labels.

Details

The menu item 'Select subset by Boolean', connected to the function selectSubset, enables the user to query by any combination of the numeric columns and labels in the whole dataset. The current data will be replaced by its newly chosen subset.

First, the user is prompted to enter a search pattern which can contain conditions that may employ most of the comparison operators common in R, i.e. < (lower than), > (greater than), <= (lower or equal to), >= (greater or equal to), = or == (equal to), != (not equal to). The character strings should be quoted. The conditions can be combined together by logical and, or and brackets.

Logical and can be expressed as .and. .AND. &

Logical or can be expressed as .or. .OR. |

Please note that at the moment no extra spaces can be handled (apart from in quoted character strings).

Value

Overwrites the data frame 'labels' and numeric matrix 'WR' by subset that fulfills the search criteria.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

regular.expressions regex

Examples

```
## Not run:
# Valid search patterns

Intrusion="Rum"
# Finds all analyses from Rum

Intrusion="Rum".and.SiO2>65
Intrusion="Rum".AND.SiO2>65
Intrusion="Rum"&SiO2>65
# All analyses from Rum with silica greater than 65
# (all three expressions are equivalent)


MgO>10&(Locality="Skye"|Locality="Islay")
# All analyses from Skye or Islay with MgO greater than 10

MgO>=10&(Locality!="Skye"&Locality!="Islay")
# All analyses from any locality except Skye and Islay with MgO greater
# or equal to 10

Locality="^S"
# All analyses from any locality whose name starts with capital S

## End(Not run)
```

---

bpplot2 *Box-Percentile Plot*

Description

Displays statistical distribution each of the variables in a data frame using a box-percentile plot *(Esty & Banfield 2003)*.

Usage

```
bpplot2(x,main="Box-Percentile Plot",sub="",xlab = "",
ylab="",log="y",col="lightgray",horizontal=FALSE,ylim = NULL,axes=TRUE,...)
```

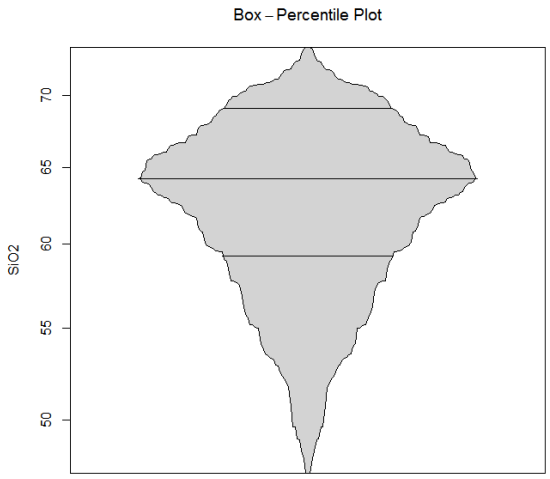
Arguments

x	data frame with the data to be plotted
main	main title for the plot

sub	sub title for the plot
xlab	label for x axis
ylab	label for y axis
log	which of the axes is to be logarithmic?
col	colour to fill the boxes
horizontal	logical, should be the orientation horizontal?
ylim	optional; limits for the y axis
axes	logical; should be the axis drawn?
...	additional plotting parameters

Details

The box-percentile plot is analogous to a boxplot but the width of the box is variable, mimicking the distribution of the given variable. As in boxplots, the median and two quartiles are marked by horizontal lines.



Value

None.

Warning

This function IS NOT Figaro-compatible. It means that the set of diagrams cannot be further edited in GCDkit (e.g. tools in "Plot editing" menu are inactive).

Author(s)

The code represents a modified function 'bpplot' from the package 'Hmisc' by Frank E Harrell Jr. (originally designed by Jeffrey Banfield). Adopted for GCDkit by Vojtěch Janoušek, <vojtech.janousek@geology.cz>.

References

Esty, WW & Banfield JD (2003) The Box-Percentile Plot. Journal of Statistical Software 8 (17)

Examples

```
data(blatna)
accessVar("blatna")

windows()
bpplot2(WR[,"K2O"],main="My box-Percentile Plot",ylab = annotate("K2O"),
    log="",col="khaki",ylim=c(3,4.5))
```

---

calc *Calculate a new variable*

Description

Calculates a single numeric variable and appends it to the data.

Usage

```
calc()
```

Details

The formula can invoke any combination of names of existing numerical columns, with the constants, brackets, arithmetic operators `+-*/^` and R functions. See `calcCore` for a correct syntax.

If the result is a vector of the length corresponding to the number of the samples in the system, the user is prompted for the name of the new data column. Unless a column with the specified name already exists or the given name is empty, the newly calculated column is appended to the data in memory ('`WR`').

Value

`results` numerical vector with the results

Modifies, if appropriate, the numeric matrix '`WR`'.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

`selectColumnLabel`.

Examples

```
## Not run:
# examples of valid formulae....
(Na2O+K2O)/CaO
Rb^2
log10(Sr)
mean(SiO2)/10

# ... but this command is in fact a simple R shell -
# meaning lots of fun for power users!
summary(Rb,na.rm=T)
cbind(SiO2/2,TiO2,Na2O+K2O)
cbind(major)
hist(SiO2,col="red")
boxplot(Rb~factor(groups))

# possibilities are endless
plot(Rb,Sr,col="blue",pch="+",xlab="Rb (ppm)",ylab="Sr (ppm)",log="xy")

## End(Not run)
```

`calcCore` *Calculation of user-defined parameters*

Description

Calculates a user-defined parameter specified by the equation.

Usage

```
calcCore(equation, where = "WR", redo = TRUE)
```

Arguments

`equation`	a text string to be evaluated.
`where`	which matrix should be used?
`redo`	logical; should be the routine called again and again?

Details

This is a core calculation function.

The expression specified by '`equation`' can involve any combination of names of existing numerical columns in the matrix '`where`', numbers (i.e. constants), arithmetic operators `+-*/^` and R functions.

The most useful of the latter are '`sqrt`' (square root), '`log`' (natural logarithm), '`log10`' (common logarithm), '`exp`' (exponential function), '`sin`', '`cos`' and '`tan`' (trigonometric functions).

Potentially useful can be also `min` (minimum), `max` (maximum), `length` (number of elements/cases), '`sum`' (sum of the elements), '`mean`' (mean of the elements), and '`prod`' (product of the elements).

However, any user-defined function can be also invoked here.

For most statistical functions, an useful parameter 'na.rm=T' can be specified. This makes the function to calculate the result from the available data only, ignoring the not determined value (see Examples).

The quotation marks in 'equation' need to preceded by a backslash. Option 'redo' specifies whether the routine should be called repeatedly until some meaningful result is obtained. Otherwise 'NA' is returned.

Value

A list of three items:

equation	equation as entered by the user
results	numeric vector with the results or NA if none can be calculated
formula	the unevaluated expression corresponding to the 'equation'

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

molecularWeight, mean

Examples

```
data(blatna)
accessVar("blatna")

calcCore("SiO2/10")

calcCore("Na2O+K2O")

calcCore("log10(Na2O+K2O)")

calcCore("SiO2/MW[\"SiO2\"]")
# dividing by the built-in molecularWeight, NB the backslashes

calcCore("length(MgO)")

calcCore("mean(MgO,na.rm=TRUE)")
# na.rm is a safety measure in case some missing values are present
# otherwise the result would be 'NA'
```

---

clr.transform *Centered-log-ratio transformation*

Description

Implementation of centred-log-ratio (clr) transformation for compositional data.

Usage

```
clr.trans(comp.data=NULL,GUI=FALSE)

pr.comp.clr(comp.data=NULL,use.cov=FALSE,scale=TRUE,GUI=FALSE)

lda.clr(comp.data=NULL,grouping=groups,GUI=FALSE)
```

Arguments

comp.data	a numerical matrix; the data to be normalized. Or just names of variables in the data matrix 'WR'.
use.cov	logical; should be the covariance matrix used instead of correlation matrix?
scale	logical; the scalings applied to each variable.
GUI	logical; is the function called from a menu (GUI)?
grouping	character or factor; grouping information for each of the samples.

Details

Compositional data - i.e., multivariate data in which all the components sum up to some constant (e.g. 1 or 100, for percentages) - are widespread in the geosciences. A typical example represent major-element analyses from whole-rock samples.

Numerous workers have argued that much of correlation in such closed datasets is spurious, due to the so-called constant sum or closure effect *(e.g., Chayes 1960; Rock 1988; Rollinson 1992, 1993).*

This effect arises from the fact that such components in the compositional datasets cannot vary independently. If one oxide, for instance SiO_2 that dominates the whole-rock analyses of many igneous rocks, increases in abundance, all other oxides must decrease. Therefore, everything must be anti-correlated with silica.

For their correct statistical treatment, compositional data have to be transformed, or 'opened'. A classic remedy to the closure effect are log-ratio transformations *(Aitchison 1986; Buccianti et al. eds 2006)*.

The functions 'clr.trans', 'pr.comp.clr' and 'lda.clr' implement the so-called centred-log-ratio (clr) transformation. Data opening in this case is done by dividing each value of a variable by the geometric mean of all the variables for that sample and then taking logarithms. It is critical of course that all the variables are expressed in the same measurement unit.

For instance, for MgO, the centred-log-ratio transformed version is given as:

$$MgO\_clr = \ln\left(\frac{C_{MgO}}{\sqrt[n]{\prod_{i=1}^{n} C_i}}\right)$$

where 'ln' is natural logarithm, 'C' concentration in wt. % of the selected variable (oxide) and the denominator a geometric mean of all variables being transformed *(e.g., Pawlowsky-Glahn & Egozcue 2006)*).

The function 'pr.comp.clr' performs principal components analysis and plots a biplot *(Gabriel, 1971; Buccianti & Peccerillo, 1999)*. The function 'lda.clr' serves for linear discriminant analysis.

Value

For `clr.trans`, a numeric matrix '`results`'. The names of components are preserved, and supplemented by a suffix '`_clr`'.

Plugin

disclosure.r

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Vladimír Kusbach, <kusbach@gmail.com>

References

Aitchison J (1986) The Statistical Analysis of Compositional Data. Methuen, New York, pp 1-416

Buccianti A, Mateu-Figueras G, Pawlowsky-Glahn V (eds) (2006) Compositional Data Analysis in the Geosciences. Geological Society London Special Publications 264: pp 1-212

Chayes F (1960) On correlation between variables of constant sum. J Geophys Res 65: 4185-4193 doi: 10.1029/JZ065i012p04185

Gabriel KR (1971) The biplot graphical display of matrices with application to principal component analysis. Biometrika 58: 453-467 doi: 10.1093/biomet/58.3.453

Greenacre, M. J. (2010). Biplots in Practice. Bilbăo: Fundación BBVA.

Pawlowsky-Glahn V, Egozcue JJ (2006) Compositional data and their analysis: an introduction. In: Buccianti A, Mateu-Figueras G, Pawlowsky-Glahn V (eds) Compositional Data Analysis in the Geosciences. Geological Society London Special Publications 264: pp 1-10 doi: 10.1144/GSL.SP.2006.264.01.01

Reimann C, Filzmoser P, Garrett R, Dutter R (2008) Statistical Data Analysis Explained: Applied Environmental Statistics with R. John Wiley & Sons, Chichester, pp 1-362

Rock NMS (1988) Numerical geology. A Source Guide, Glossary and Selective Bibliography to Geological Uses of Computers and Statistics. Lecture Notes in Earth Sciences 18, Springer, Berlin, pp 1-427 doi: 10.1007/BFb0045143

Rollinson HR (1992) Another look at the constant sum problem in geochemistry. Mineral Mag 56: 469-475 doi: 10.1180/minmag.1992.056.385.03

Rollinson HR (1993) Using Geochemical Data: Evaluation, Presentation, Interpretation. Longman, London, pp 1-352 doi: 10.4324/9781315845548

van den Boogaart KG, Tolosana-Delgado R (2008) "compositions": a unified R package to analyze compositional data. Comput Geosci 34: 320-338 doi: 10.1016/j.cageo.2006.11.017

van den Boogaart KG, Tolosana-Delgado R (2013) Analyzing Compositional Data with R. Springer, Berlin, pp 1-258

Venables WN, Ripley BD (1999) Modern Applied Statistics with S-Plus. Springer, Berlin. doi: 10.1007/9781475731217

See Also

prComp princomp lda

See *Reimann et al. (2008)* with *van den Boogaart and Tolosana-Delgado (2013)* for further details and *van den Boogaart and Tolosana-Delgado (2008)* for implementation of a comprehensive R library dealing with compositional data.

Examples

```
data(sazava)
accessVar("sazava")

# Centered-log-ratio transformation
ox<-c("SiO2","Al2O3","FeOt","MgO","CaO")
clr.trans(ox)
addResults() # Needed to append the clr-transformed data to the matrix 'WR'

multiple(x="SiO2_clr", y="Al2O3_clr,FeOt_clr,MgO_clr,CaO_clr")
plateCex(2)
plateCexLab(1.3)

# Principal components on basis of clr-transformed data
pr.comp.clr()

pr.comp.clr("SiO2,TiO2,Al2O3,MgO,CaO")
```

`cluster` *Statistics: Hierarchical clustering*

Description

Hierarchical cluster analysis on a set of dissimilarities.

Usage

```
cluster(x=NULL,elems="SiO2,TiO2,Al2O3,FeOt,MnO,MgO,CaO,Na2O,K2O",label.by=1,
    method="average",id.clusters=TRUE)
```

Arguments

`x`	numerical matrix with compositional data.
`elems`	numerical columns to be used for cluster analysis, typically major elements.
`method`	the agglomeration method to be employed. This should be one of (or an unambiguous abbreviation thereof): `'ward.D'`,`'single'`,`'complete'`, `'average'`,`'mcquitty'`,`'median'`,`'centroid'`.
`label.by`	numeric; names for each of the cases (samples).
`id.clusters`	logical; should be individual clusters indentified interactively?

Details

Even though a list of major elements is assumed as a default, different variables can be specified. In GUI, this is done by the function '`selectColumnsLabels`'. Moreover, samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSamples` for details.

The user can also specify a label for the individual samples, default are their names (`label.by` = 1). For `label.by` > 1, the number indicates a sequence number of the column in the `labels` + 1.

In GUI, when `id.cluster` is `TRUE`, individual clusters can be identified after the dendrogram is drawn. For each sample falling into the given group, specified information (e.g. Locality, Rock Type and/or Author) can be printed.

For further details on the clustering algorithm, see the R manual entry of '`hclust`'.

Value

None.

Warning

Names of existing numeric data columns and not formulae involving these can be handled at this stage. As only complete cases are used for the cluster analysis, missing values are replaced by 0.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'hclust'

Examples

```
data(sazava)
accessVar("sazava")

cluster(WR,elems = LILE,method = "ward")

cluster(WR,elems = LILE,method = "ward.D",label.by=2,id.clusters=FALSE)
```

---

combined *Sample mineral compositions of K-feldspars, clinopyroxenes and garnets adopted from Richard (1995).*

Description

This is a sample mineral dataset with compositions of K-feldspars, clinopyroxenes and garnets adopted from old, now disused Windows software MinPet by *L. R. Richard (1995)*.

Usage

```
data("combined")
```

Format

A data frame containing 32 observations.

Source

Richard LR (1995) MinPet: Mineralogical and Petrological Data Processing System, Version 2.02. MinPet Geological Software, Québec, Canada.

See Also

data, accessVar

Examples

```
data(combined)
accessVar(combined)
```

`contourAll` *Outlining the whole dataset in a binary plot*

Description

These functions outline the whole dataset on a binary plot. Implemented methods are the convex hull or contours. This can be useful for a quick appreciation of the data distribution, e.g. in classification diagrams.

Usage

```
chullAll(border=NULL, fill=FALSE, ...)

contourAll(cont.levels = c(0.25,0.50,0.75), n = 20,
    border = NULL, fill = FALSE, fade.away = TRUE,...)
```

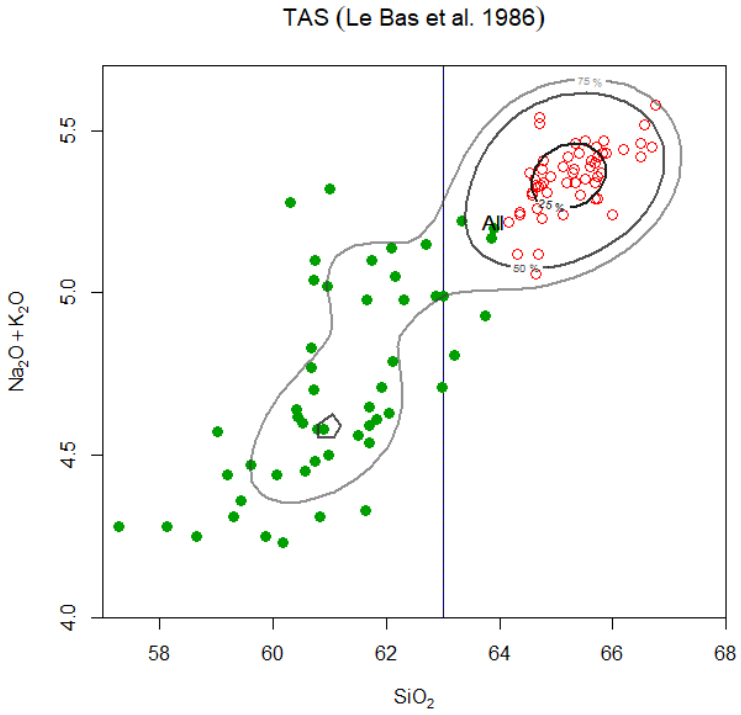
Arguments

`border`	outline colour.
`fill`	logical; should be the polygon filled by the border colour?
`cont.levels`	values (0-1) where contours are to be drawn.
`n`	density of the grid.
`fade.away`	logical; shall the colours of individual contours fade away?
`...`	additional parameters to the functions contour and polygon, respectively.

Details

If not specified, the colours are selected as the most frequently occurring one among the samples within each group.

For the function *contourAll*, contours are drawn based on percentage of the whole population, based on the kernel density estimation. Their smoothness (vs. speed of computation) is determined by the parameter `n`. The individual contours can be made increasingly more transparent, as controlled by the parameter `fade.away`.



Value

Returns (invisibly) a list with two components:

z	Values of estimated densities for each of the `cont.levels`.
kde	A matrix of the estimated density: rows correspond to the value of `x`, columns to the value of `y`.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz> and

Jean-François Moyen <jfmoyen@gmail.com>

See Also

chull, optimize kde2d contour, polygon figaro contourGroups chullGroups

Examples

```
data(atacazo)
accessVar("atacazo")

binary("SiO2","Na2O+K2O")
figUser(xlim=c(57,68),ylim=c(4,5.7))
contourAll(n=10,border="gray",fade.away=FALSE)

figRedraw()
out<-contourAll(n=50,border="darkgray",fade.away=TRUE)
```

```
windows()
xlab<-.fig.deeval(sheet$demo$call$xlab)
ylab<-.fig.deeval(sheet$demo$call$ylab)
persp(out[["All"]]$kde,xlab=xlab,ylab=ylab,main="All")

binary("SiO2","K2O")
figUser(xlim=c(55,70),ylim=c(0.5,1.5))
chullAll(border="darkgray")

chullAll(border="gray",fill=TRUE)
contourAll(cont.levels=c(0.9,0.5,0.1),border="darkblue",fade.away=FALSE,lty="dashed")
```

`contourGroups` *Outlining individual groups of samples in a binary plot*

Description

These functions outline the individual clusters of data (groups by default) on a binary plot. Implemented methods are the convex hull or contours. This can be useful for a quick appreciation of the data distribution, e.g. in classification diagrams.

Usage

```
chullGroups(clusters = groups, border = NULL, fill = FALSE,...)

contourGroups(clusters = groups, cont.levels = c(0.25,0.50,0.75), n = 20,
    border = NULL, fill = FALSE, fade.away = TRUE,...)
```

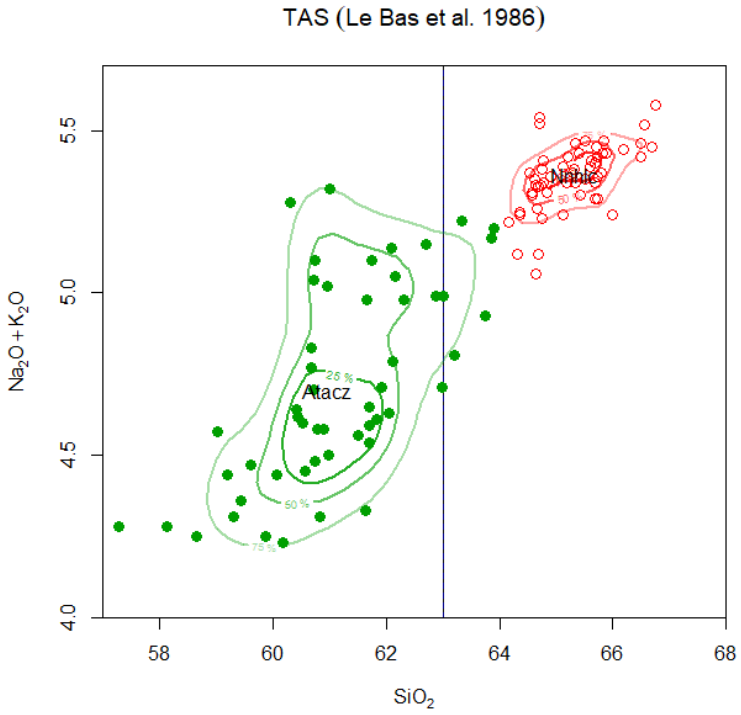
Arguments

`clusters`	grouping information for each of the samples.
`border`	outline colours.
`fill`	logical; should be the polygons filled by the border colour?
`cont.levels`	values (0-1) where contours are to be drawn.
`n`	density of the grid.
`fade.away`	logical; shall the colours of individual contours fade away?
`...`	additional parameters to the functions contour and polygon, respectively.

Details

If not specified, the colours are selected as the most frequently occurring one among the samples within each group.

For the function *contourGroups*, contours are drawn based on percentage of the whole population, based on the kernel density estimation. Their smoothness (vs. speed of computation) is determined by the parameter `n`. The individual contours can be made increasingly more transparent, as controlled by the parameter `fade.away`.



Value

Returns (invisibly) a list with two components:

z	Values of estimated densities for each of the `cont.levels`.
kde	A matrix of the estimated density: rows correspond to the value of `x`, columns to the value of `y`.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz> and

Jean-François Moyen <jfmoyen@gmail.com>

See Also

chull, optimize kde2d contour, polygon figaro contourAll chullAll

Examples

```
data(atacazo)
accessVar("atacazo")
groupsByLabel("Volcano")

binary("SiO2","Na2O+K2O")
figUser(xlim=c(57,68),ylim=c(4,5.7))
contourGroups(n=10,fade.away=FALSE)

figRedraw()
out<-contourGroups(n=50,fade.away=TRUE)
```

```
windows()
xlab<-.fig.deeval(sheet$demo$call$xlab)
ylab<-.fig.deeval(sheet$demo$call$ylab)
persp(out[["Atacazo"]]$kde,xlab=xlab,ylab=ylab,main="Atacazo")

binary("SiO2","K2O")
figUser(xlim=c(55,70),ylim=c(0.5,1.5))
chullGroups()

chullGroups(fill=TRUE)

figRedraw()
contourGroups(cont.levels=c(0.9,0.5,0.1),border="darkgray",fade.away=FALSE,lty="dashed")
```

`crosstab` *Cross table of labels*

Description

Prints a cross table (contingency table) for 1-3 labels.

Usage

```
crosstab(plot = TRUE)
```

Arguments

`plot`	logical; should be also a barplot plotted?

Details

This command prints a frequency distribution (for a single label) or a contingency table (for 2-3 labels) useful for inspection of the data structure. Optionally a barplot is plotted (for 1-2 labels).

Just press Enter (enter an empty field), when the desired number of variables is reached.

Value

`results`	the frequency/contingency table

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

cutMy *Groups by numerical variable*

Description

Grouping the data according to the interval of a single numerical variable it falls into.

Usage

```
cutMy(where=NULL,int=NULL,int.lab=NULL,na.lab="Unclassified")
```

Arguments

where	Numeric data column in 'WR' - the basis of the classification.
int	Boundaries of intervals.
int.lab	Labels for the intervals
na.lab	Labels for samples that cannot be classified

Details

The numeric data column is selected using the function '`selectColumnLabel`'.

After this is done, the user is prompted to enter a comma-delimited list or at least one break point defining the intervals. The default includes the mean, that will be automatically supplemented by minimum and maximum (i.e. two intervals).

Then the names of the individual groups are to be specified; values out of range are automatically labeled as `'Unclassified'`. The vector containing the information on the current groups can be appended to the data frame '`labels`'.

Value

groups	character vector: the grouping information
grouping	If the new column was appended the data frame `labels`, sequence number of this column; if not appended, though, this variable is set to -100.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

`cut`

Examples

```
data(sazava)
accessVar(sazava)

# Two groups defined based on SiO2
cutMy("SiO2",int=c(0,60,100),c("Low-Si","High-Si"))
```

Edit labels *Edit labels*

Description

Simultaneous editing of all labels using a spreadsheet-like interface.

Usage

```
editLabels()
```

Arguments

none.

Details

The function invokes a spreadsheet-like interface that enables the user to edit the labels for individual samples. When all the desired changes have been performed, close button is to be clicked.

Value

Returns the corrected version of the data frame 'labels'.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'data.entry'

Edit numeric data *Edit numeric data*

Description

Simultaneous editing of all numeric data using a spreadsheet-like interface.

Usage

```
editData(x=WR)
```

Arguments

x	data frame/numeric matrix to be edited; default is 'WR', i.e. numeric data

Details

The function invokes a spreadsheet-like interface that enables the user to edit the numeric data for individual samples. When all the desired changes have been performed, close button is to be clicked.

The system then performs some recalculations as if the data set was loaded from the disc afresh (calling 'Gcdkit.r').

Value

Returns the corrected version of the numeric matrix 'WR'.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'data.entry'

---

editLabFactor *Edit label as factor*

Description

Global replacement each of the discrete values (levels) for a selected label.

Usage

```
editLabFactor()
```

Details

The label to be edited is selected using the function 'selectColumnLabel'.

Then the function invokes a spreadsheet-like interface that enables the user to overwrite directly any of the discrete values for the a given label, in the R jargon called *levels*. When all the desired changes have been performed, close button is to be clicked.

Value

Returns the corrected version of the data frame labels.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'data.entry'

`Export to Access` *Export to Access*

Description

This function serves for exporting the specified data into *.MDB (MS Access) format (via the ODBC interface).

Usage

```
accessExport(what=cbind(labels, WR), tablename=NULL,
transpose=FALSE,dec.places=NULL)
```

Arguments

`what`	a matrix, data frame or a list
`tablename`	name of the data table
`transpose`	logical; transpose the data?
`dec.places`	numeric; number of decimal places

Details

The function `accessExport` outputs the specified data via Microsoft's ODBC interface, taking an advantage of the library `RODBC`. Unlike for the function '`excelExport`', ODBC makes possible opening a new file.

If the argument `what` is a matrix or data frame, the name of the table can be specified using the optional parameter `tablename`.

For a list, several tables are created, their number and names corresponding to the items present.

Value

None.

Warning

This function is not available on 64-bit systems!

Author(s)

The RODBC package was written by Brian Ripley.

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'`excelExport`' '`excel2007Export`' '`dbfExport`'

Examples

```
    ## Not run:
        accessExport(results) # Saves the last calculated results

## End(Not run)
```

`Export to DBF` *Export to DBF*

Description

This function serves for exporting the specified data into DBF (dBase III) format (using the function '`write.dbf`' of the package '`foreign`').

Usage

```
dbfExport(what=cbind(labels,WR), transpose=FALSE)
```

Arguments

`what`	a matrix or data frame
`transpose`	logical; transpose the data frame?

Details

The function `dbfExport` outputs the specified data. Note that it cannot handle lists.

Value

None.

Warning

This function is not available on 64-bit systems!

Author(s)

The RODBC package was written by Brian Ripley.

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'`write.dbf`' '`excelExport`' '`excel2007Export`' '`accessExport`'

Examples

```
    ## Not run:
        dbfExport(results) # Saves the last calculated results

## End(Not run)
```

Export to Excel *Export to Excel*

Description

This function serves for exporting the specified data into XLS or XLSX (MS Excel) formats (via the ODBC interface).

Usage

```
excelExport(what=cbind(labels, WR), tablename =NULL,
transpose=FALSE, dec.places=NULL)

excel2007Export(what=cbind(labels, WR), tablename =NULL,
transpose=FALSE, dec.places=NULL)
```

Arguments

what	a matrix, data frame or a list
tablename	name of the data sheet
transpose	logical; transpose the data?
dec.places	numeric; number of decimal places

Details

The functions excelExport and excel2007Export output the specified data via Microsoft's ODBC interface, taking an advantage of the library 'RODBC'.

If the argument 'what' is a matrix or data frame, the name of the sheet can be specified using the optional parameter 'tablename'.

For a list, several sheets are attached, their number and names corresponding to the items present.

Value

None.

Warning

These functions are not available on 64-bit systems!

Author(s)

The RODBC package was written by Brian Ripley.

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'accessExport' 'dbfExport'

Examples

```
## Not run:
    excelExport(results) # Saves the last calculated results in XLS format

  excel2007Export(results) # Saves the last calculated results in XLSX (or XLS) format

## End(Not run)
```

`Export to HTML tables` *Exporting data or results to HTML tables*

Description

Outputs the specified data with (optional) labels into HTML. This format is useful for importing into spreadsheets, word processors or publishing on the WWW.

Usage

```
HTMLTableMain(what, digits=2, desc=NULL, title=" ", sum.up=FALSE,
open=TRUE, close=TRUE, browse=TRUE,
filename=paste(data.dir,"R2HTML/htmltable",sep="/"),
rotate=FALSE)

HTMLtableOrdered(what, which=rownames(what), labs=labels, digits=2,
desc=NULL, title=" ", sum.up=FALSE, key1=NULL, key1descending=FALSE,
key2=NULL, filename=paste(data.dir,"R2HTML/htmltable",sep="/"),
split.by=25, rotate=TRUE)

HTMLTableWR(filename="htmltable")

HTMLTableWR.Jupyter(digits=3, desc=NULL, title=" ", sum.up=TRUE,
rotate=FALSE)

HTMLTableResults(filename="htmltable")

HTMLTableResults.Jupyter(digits=3, desc=NULL, title=" ", sum.up=TRUE,
rotate=FALSE)
```

Arguments

`what`	numeric matrix; data to be exported
`digits`	required precision
`desc`	name of the columns within 'labels' to be attached to the table
`title`	main title
`sum.up`	logical; should be a sum calculated?
`open`	logical; should be opened a new HTML file?
`close`	logical; should be the HTML file closed when finished?
`browse`	logical; should be the HTML file finally opened in the default browser?
`filename`	optional name for the file produced

rotate	logical, should be the table transposed, with samples in columns and variables in rows?
which	(optional) sample names in numeric matrix 'what' for the output
labs	name of variable with textual labels
key1	is a variable in numeric matrix 'what'
key1descending	logical; should be the 'key1' sorted in a descending order?
key2	is a grouping information (name of a column in 'labs' or a character vector)
split.by	maximal number of data columns per page

Details

HTMLTableWR and HTMLTableResults are GUI front ends to HTMLTableMain, the former enabling the user to choose samples (rows) and columns for the output using the searching mechanisms common in the GCDkit.

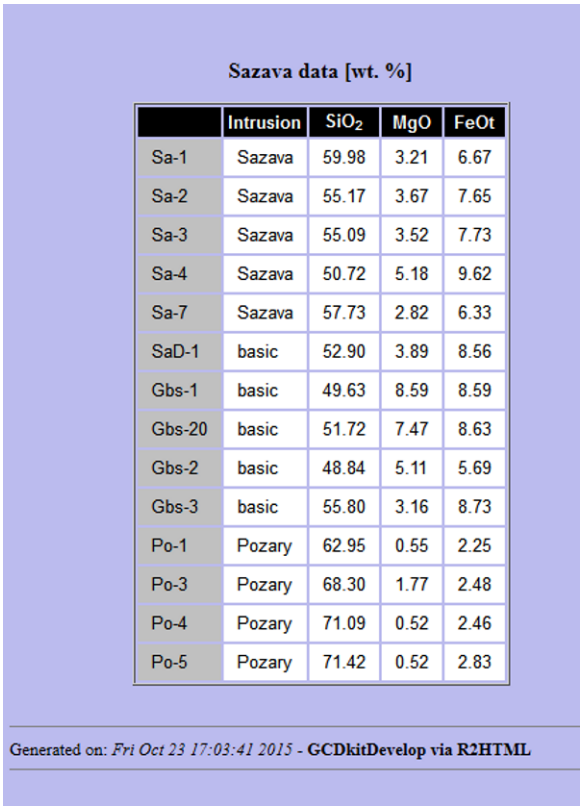
HTMLTableWR outputs the numeric data (with optional labels and sum) stored in the data matrix 'WR'. The function HTMLTableWR.Jupyter provides the same functionality for Jupyter notebooks.

HTMLtableOrdered also outputs the numeric data stored in the numeric matrix specified by parameter 'what'. Optional argument 'which' gives the list of sample names (rows) in the matrix to be saved. The data are first sorted based on 'key2', which typically gives a grouping information (name of a column in 'labs'). Within each of the groups, the data are further sorted based on the numeric variable 'key1'. See example.

HTMLTableResults outputs the results of the most recent calculation (with optional labels and sum) as stored in the variable 'results'. The function HTMLTableResults.Jupyter provides the same functionality for Jupyter notebooks.

This function attempts to format sub- and superscripts in the names of variables.

The created file 'filename' is placed in the system temporary folder; when finished, it is previewed in a browser. The appearance for the table is determined by the cascade style file R2HTML.css in the *GCDkit* main subdirectory.



Sazava data [wt. %]

	Intrusion	SiO_2	MgO	FeOt
Sa-1	Sazava	59.98	3.21	6.67
Sa-2	Sazava	55.17	3.67	7.65
Sa-3	Sazava	55.09	3.52	7.73
Sa-4	Sazava	50.72	5.18	9.62
Sa-7	Sazava	57.73	2.82	6.33
SaD-1	basic	52.90	3.89	8.56
Gbs-1	basic	49.63	8.59	8.59
Gbs-20	basic	51.72	7.47	8.63
Gbs-2	basic	48.84	5.11	5.69
Gbs-3	basic	55.80	3.16	8.73
Po-1	Pozary	62.95	0.55	2.25
Po-3	Pozary	68.30	1.77	2.48
Po-4	Pozary	71.09	0.52	2.46
Po-5	Pozary	71.42	0.52	2.83

Generated on: *Fri Oct 23 17:03:41 2015* - **GCDkitDevelop via R2HTML**

Value

None.

Warning

All these functions require the 'R2HTML' library. It must be downloaded from the CRAN and properly installed. Their presence is checked before the code is executed.

Jupyter versions of the functions `HTMLTableWR.Jupyter` and `HTMLTableResults.Jupyter` require a correctly configured IR kernel connection, including the 'IRdisplay' library.

Author(s)

The R2HTML package was written by Eric Lecoutre.

The IRdisplay package was written by Thomas Kluyver, Philipp Angerer and Jan Schulz.

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(sazava)
accessVar("sazava")

HTMLTableMain(WR[,c("SiO2","MgO","FeOt")],digits=2,desc="Intrusion",
    title="Sazava [wt.%]")

HTMLtableOrdered(WR[,LILE],digits=1,key1="SiO2",key2="Intrusion",
    title="Large Ion Lithophile Elements (ppm)",split.by=3)
```

feldspars *Selected feldspar analyses*

Description

This data set gives the selected chemical analyses of feldspars from *Deer et al. (2013)*.

Usage

```
data(feldspars)
```

Format

A data frame containing 20 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 270, 299 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(feldspars)
accessVar("feldspars")
print(WR)
```

figAdd *Plot editing: Add*

Description

These functions enable adding new components to Figaro-compatible plots.

Usage

```
figTicks(major=-0.5, minor=0.25, xmjr=NULL, xmin=NULL, ymjr=NULL, ymin=NULL)

figGrid(x.int=NULL, y.int=NULL, lty="dotted", col="gray30",GUI=FALSE)

figLegend(x=NULL,y=NULL,bg="#FFFFFFAA",...)

figAddText()

figAddArrow()

figAddBox()

figAddFit(lty="solid",col="black",by.group=FALSE)

figAddCurve(equation=NULL)
```

Arguments

major	length of the major tick marks.
minor	length of the minor tick marks.
xmjr, ymjr	intervals for the major tick marks.
xmin, ymin	intervals for the minor tick marks.
x.int	intervals for the grid, x axis component.
y.int	intervals for the grid, y axis component.
GUI	logical; is the function called from GUI?
x,y	coordinates for the legend.
bg	background for the legend.
...	additional parameters to the plotting function. See showLegend and figOverplot, respectively.
lty	line type.
col	plotting colour.
by.group	logical; should be the linear regression performed by groups?
equation	text; equation expressed as a function of x; see curve.

Details

'figTicks' adds major and minor tick marks for the x and y axes. Their length is specified as a fraction of the height of a line of text. Negative numbers imply outward and positive inward pointing ticks. The user is prompted for four numbers separated by commas, xmjr, xmin, ymjr, ymin. These specify the intervals of major and minor ticks for x and y axes, respectively. Not implemented to logarithmic plots and spiderplots yet.

'figGrid' adds grid lines for x and/or y axes.

'figLegend' adds legend(s) on specified location. See legend and showLegend for further details.

'figAddText' adds text on specified location. The parameters are the text style ('n' = normal, 'b' = bold, 'i' = italic and 'bi' = bold italic), colour and relative size.

'figAddArrow' adds arrow on specified location. The parameters are colour and line style ('solid', 'dashed', 'dotted' and 'dotdash').

'figAddBox' adds box on specified location (click bottom left and then top right corner).

'figAddFit' adds either a single least-squares fit to all data, or several fit lines, for each of the groups separately. The parameters are colour and line style ('solid', 'dashed', 'dotted' and 'dotdash'). If using with GUI, the equation of each fit line is plotted at the user-defined location.

'figAddCurve' adds a curve, specified as a function of variable 'x'. The parameters are colour and line style ('solid', 'dashed', 'dotted' and 'dotdash').

The colours can be specified both by their code (see table under menu 'Data handling|Show available colours') or R name (see Examples).

The additional two menu items, available for binary and ternary plots, allow adding contours or convex hulls outlining individual groups of data. See contourGroups and chullGroups.

Value

None.

Warning

These functions serve to adding some extra components/annotations immediately before the graph is printed/exported. Note that all these user-defined components added via 'Plot editing: Add' will be lost upon redrawing, zooming ....

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

par showColours colours

figaro figAddReservoirs

contourGroups chullGroups legend showLegend

curve

Examples

```
data(blatna)
accessVar("blatna")
setCex(1.5)

## figTicks and figGrid
binary("Zr/Nb","Ba/La")
figTicks(major=-0.5, minor=0.25,10,1,10,1)
figGrid(,5,col="darkblue") # just y axis (second parameter)

figRedraw()
figGrid(2,5,col="darkblue")

## figLegend
groupsByLabel("Suite")
figLegend(x="bottomleft",bg="#AAAAAAAA") # Semitransparent
```

figAddReservoirs *Plot editing: Add Reservoirs*

Description

This functions overplots data from some geochemical reservoirs to Figaro-compatible plots.

Usage

```
figAddReservoirs(autoscale=FALSE, var.name=NULL, sample.names=NULL,
reserv.condition=NULL, labs=NULL, pch="*", col="darkred", cex=1, type="p",...)
```

Arguments

autoscale	logical; should be the scaling changed so that all the plotted data fit in?
var.name	text; either 'reservoirs.data', 'idealmins.data'or a name of a global variable. See Details.
sample.names	character vector; names of reservoirs, ideal minerals or samples to be plotted.
reserv.condition	text; regular expression specifying reservoirs compositions of which are to be plotted.
labs	text; optional abbreviated labels for the individual reservoirs
pch	plotting symbol.
col	plotting colour.
cex	numeric; relative size of the plotting symbol.
type	character; plot type; see plot.default.
...	further arguments of the respective plotting functions.

Details

Please note that the function 'figAddReservoirs' in the current version of *GCDkit.Mineral* is for internal use only.

Value

A numeric matrix with the overplotted analyses from the reference dataset.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

par showColours colours

figaro

contourGroups chullGroups legend showLegend

curve

Examples

```
data(blatna)
accessVar("blatna")
setCex(1.5)

## figAddReservoirs
# binary plots
binary("Zr/Nb","Ba/La")
# Sun & McDonough mantle reservoirs, Taylor & McLennan 1995 Upper and Lower Crust
reserv<-c("(MORB|OIB)..Sun","Upper Continental Crust..Taylor","Lower Continental Crust..Taylor")
reserv.names<-c("NMORB","EMORB","OIB","UCC","LCC")
figAddReservoirs(autoscale=TRUE,var.name="reservoirs.data",reserv.condition=reserv,
    labs=reserv.names)

binary("SiO2","K2O")
figAddReservoirs(TRUE,var.name="idealmins.data",sample.names=c("Or","Bt","Ph"))

# ternary plots
plateExtract("Wood",1)
reserv<-c("(MORB|OIB)..Sun","Upper Continental Crust..Taylor","Lower Continental Crust..Taylor")
reserv.names<-c("NMORB","EMORB","OIB","UCC","LCC")
figAddReservoirs(TRUE,"reservoirs.data",reserv.condition=reserv,labs=reserv.names)

ternary("SiO2/10","MgO","FeOt")
figAddReservoirs(var.name="idealmins.data",sample.names=c("Or","Bt","Ph"))

# spider plots
spider(WR,"NMORB..Sun",field=TRUE,colour="gray",field.colour=TRUE,ymin=0.1,ymax=1000)
figAddReservoirs(var.name="reservoirs.data",reserv.condition="Continental Crust",
    autoscale=TRUE,col=c("red","black","darkblue"),pch=1:3)

# Calculate Rayleigh-type fractionation trend
ff<-seq(1,0.1,-0.1) # F, amount of melt left
x<-80*ff^(1.2-1)    # cL for three elements, arbitrary D of 1.2, 2.0 and 1.3
y<-550*ff^(2.0-1)
z<-1000*ff^(1.3-1)
my.trend<-cbind(x,y,z)
colnames(my.trend)<-c("Rb","Sr","Ba")
rownames(my.trend)<-ff

# By default, the overplotted information is added permanently
binary("Rb","Sr",log="xy")
figAddReservoirs(var.name="my.trend",pch="+",col="blue",autoscale=TRUE,type="o",
    labs=rownames(my.trend))
figXlim(c(10,500))

# But this is controlled by the argument just.draw
binary("Rb","Sr",log="xy")
```

```
figAddReservoirs(var.name="my.trend",pch="+",col="red",autoscale=TRUE,type="o",
    labs=rownames(my.trend),just.draw=TRUE)
figRedraw()
```

---

figaro.identify *Plot editing: Identification of plotted symbols*

Description

These functions allow the user to identify points in Figaro-compatible plots.

Usage

```
figIdentify()

highlightSelection()
```

Details

'figIdentify' identifies points closest to a mouse pointer, if a mouse button is pressed. For binary and ternary plots, sample names are plotted; for spider plots the function prints the sample name, concentration of the given element (in ppm) and highlights the whole pattern. The identification is terminated by pressing the right button and selecting 'Stop' from the menu.

'highlightSelection' allows the selected analyses to be highlighted. The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSubset for details.

If the search results are empty or embrace all samples, the user is given a chance to select the samples from the list of their names. Press Ctrl+click to select multiple ones.

For binary and ternary plots, Press Esc in the Console window to stop the points blinking. In spider plots are shown overall ranges of normalized concentrations (by a gray field) with superimposed patterns for selected samples.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

and Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

identify selectSubset 'figaro'

Examples

```
## Not run:
    data(sazava)
    accessVar("sazava")

    binary("SiO2","MgO")
    figIdentify()

    highlightSelection()

## End(Not run)
```

figCol *Plot editing: Colours*

Description

These functions enable altering colours for titles or all plotting symbols in Figaro-compatible plots.

Usage

```
figCol(col=NULL)

figColMain(col=NULL)

figColSub(col=NULL)

figBw()
```

Arguments

col	colour specification

Details

The colours can be specified both by their code (see table under *Data handling|Show available colours*) or R name (see Examples).

figBw sets the whole plot (main title and subtitle, axes and plotting symbols) in black and white, making them ready for printing/exporting.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

& Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'showColours' 'colours' 'figaro'

Examples

```
data(blatna)
accessVar("blatna")

colours() # prints the list of available colour names

# Binary plot
binary("SiO2","Na2O+K2O",main="My TAS diagram")
figSub(txt="from Blatná suite")
figCol(col="green")
figColMain(col="red")
figColSub(col="blue")

figBw()
```

figEdit *Plot editing: Changing titles and axis labels*

Description

These functions enable altering titles and axis labels of binary (figXlab,figYlab) and ternary (figAlab,figBlab,figClab), Figaro-compatible plots.

Usage

```
figMain(txt=NULL)

figSub(txt=NULL)

figXlab(txt=NULL)

figYlab(txt=NULL)

figAlab(txt=NULL)

figBlab(txt=NULL)

figClab(txt=NULL)
```

Arguments

txt	text

Details

If specified, the parameter txt will be passed to the function 'annotate' to guess the correct reformatting to sub- and superscripts for production of "publication quality" plots. Otherwise, the current value (titles or labels for axes/apices) are edited.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

and Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'annotate'

'figaro'

Examples

```
data(sazava)
accessVar("sazava")

binary("SiO2","Na2O+K2O")
figMain(txt="My TAS diagram")
figSub(txt="test")
```

```
figXlab(txt="Silica")
figYlab(txt="Total alkalis")
```

---

figGbo *Defining groups on Figaro-compatible plots*

Description

Interactive definition of groups on any Figaro-compatible plot.

Usage

```
figGbo(x.tol = 0, y.tol = 0, max.points = 100, max.polygons = 25)
```

Arguments

x.tol, y.tol	tolerance for the automatic closing of polygons.
max.points	maximum number of vertices for a single polygon.
max.polygons	maximum number of polygons.

Details

Each of the groups is defined by clicking vertices of a polygon with the corresponding data points. The polygons are closed automatically. To finish, right click anywhere on the plot and select 'Stop'. The groups are numbered consecutively, points falling into two or more fields form extra groups, as do unclassified samples.

Author(s)

Vojtech Erban, <erban@sopky.cz> & Vojtech Janousek, <vojtech.janousek@geology.cz>

See Also

'figaro'

---

figMulti *Figaro: Multiple plot by groups*

Description

Displays multiple plots, for each of the groups one, based on a most recently plotted Figaro-compatible template. For spiderplots, the colour field denotes the total variation with the whole dataset.

Usage

```
figMulti(x=x.data,y=y.data,nrow=NULL,ncol=NULL,xlab=sheet$demo$call$xlab,
    ylab=sheet$demo$call$ylab,pch=NULL,col=NULL,
    cex = NULL,plot.symb=NULL,shaded.col="gray",title=NULL,...)
```

Arguments

x, y	data to be plotted
nrow, ncol	dimensions of the plots' matrix
xlab, ylab	labels for the axes
pch	plotting symbols (except for spiders)
col	plotting colours (except for spiders)
cex	relative size of the plotting symbols (except for spiders)
plot.symb	logical, spiders. Shall be shown also plotting symbols or just lines?
shaded.col	(spiders) Colour for the field portraying the overall variability in the dataset.
title	optional title for the whole plate. If not provided, it is taken from the title of the Figaro template.
...	any additional graphical parameters

Note

This function uses the plates concept. The individual plots can be selected and their properties/appearance changed as if they were stand alone Figaro-compatible plots. See Plate, Plate editing and figaro for details.

Details

The function can handle any Figaro-compatible plots, including binary, ternary or spiderplots. For classification plots, it may be advantageous to switch off the field names using the function 'figRemove' (see the figure below as well as the Examples).

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

and Colin M. Farrow, <colinfarrow537@gmail.com>

See Also

figaro, Plate, Plate editing binary, ternary

Examples

```
data(combined)
accessVar("combined")
groupsByLabel("Mineral")
# Note that groups should have been defined before running these examples.

# Original wt.% data in the whole dataset
binary("SiO2","Al2O3")
figMulti(col="black",pch="*",cex=2)

# Binary plot apfu, two pyroxene groups defined based on Al2O3
selectSubset("Mineral='PYR'")
cutMy("Al2O3",int=c(0,3,100),c("Low-Al","High-Al"))
.callGCDkit("binary('Si','Al')",mineral="clinopyroxene",what="recalc")
figYlim(c(0,0.2))
figMulti(col="black",pch="*",cex=2)
```

```
selectAll()

# ternary plot apfu, two feldspar groups defined based on K2O
selectSubset("Mineral='FEL'")
cutMy("K2O",int=c(0,2,100),c("Low-K","High-K"))
.callGCDkit("ternary('K','Ca','Na')",mineral="feldspar",what="recalc")
figMulti()
plateUser(pch=6,las=3,cex=2.5,col="darkred",cex.axis=1.5,cex.lab=1.8)
selectAll()
```

`figOverplot` *Overplotting data onto pre-existing binary, ternary or spider plots*

Description

This function allows overplotting new data points onto Figaro-compatible binary or ternary plots, or patterns onto spiderplots. It is most useful in adding selected data from typical geochemical reservoirs (e.g., Upper Continental Crust, MORB ...), ideal mineral compositions, results of petrogenetic modelling or just another dataset used for comparison (any of these will be henceforth referred to as a reference dataset).

Usage

```
figOverplot(var.name, mat=NULL, sample.names=NULL, condition=NULL,
    labs=NULL, autoscale=FALSE, pch="*", col="darkred", cex=1,
    type="p", just.draw = FALSE,overplotDataset = FALSE,...)
```

Arguments

`var.name`	either 'reservoirs.data', 'idealmins.data' or a quoted name of a global variable.
`mat`	matrix with data for all reservoirs available for overplotting. Meant mainly for internal use of the *GCDkit* system.
`sample.names`	character vector; list of names of desired reservoirs, ideal minerals or samples in the reference dataset to be overplotted.
`condition`	text; regular expression specifying names of desired reservoirs, ideal minerals or samples in the reference dataset.
`labs`	text; optional (typically abbreviated) labels for the overplotted data from the reference dataset.
`autoscale`	logical; should be the scaling changed so that all the plotted analyses fit in?
`pch`	plotting symbol(s) for the reference dataset.
`col`	plotting colour(s) for the reference dataset.
`cex`	numeric; relative size of the plotting symbol(s) for the reference dataset.
`type`	character; plot type; see plot.default. For obvious reasons, not implemented for spiderplots.
`just.draw`	logical; if `FALSE`, the overplotted bit is added permanently, i.e. the Figaro template is also affected.
`overplotDataset`	logical; for internal use by the system only.
`...`	additional parameters to the underlying plotting function(s). See Details.

Details

If called directly, the function is employed to overplot data from a reference dataset, either real-world data or a numeric matrix spanning, for instance, from petrogenetic modelling. The data originate from a two-dimensional variable in the global environment, whose name is provided via the obligatory argument '`var.name`'.

Argument '`mat`' is meant for internal use by the system and does not need to be specified by the user as the data frame/matrix `mat` is generated automatically by the function '`figOverplot`'.

In both cases, the selection from the numeric matrix or dataframe '`mat`' is based on a list of desired '`sample.names`' or on a regular expression yielding their subset ('`condition`'). Of course, from this selection, only analyses with data sufficient to be plotted on the current diagram are used.

If neither '`sample.names`' nor '`condition`' is provided, all samples are shown.

For plotting are used functions 'points', 'triplotadd' and spider for binary plots, ternary plots and spiderplots, respectively. Argument '...' can supply additional parameters to these low-level plotting functions.

Optional parameter '`labs`' can specify alternative, typically abbreviated textual labels to the points plotted.

Logical argument '`autoscale`' determines whether the plot should be rescaled to accommodate both the original data points and the reference dataset. Clearly, it does not make sense for a ternary plot.

By default, the overplotted information is added permanently but this behaviour is controlled by the argument `just.draw`.

Value

A numeric matrix with the overplotted analyses from the reference dataset.

Note

Within the *GCDkit* system, this function is invoked by 'figAddReservoirs' to overplot selected compositions from typical geochemical reservoirs (system file '`reservoirs.data`') or chemistries of ideal minerals (system file '`idealmins.data`').

Warning

If `just.draw=FALSE`, the points for the reference dataset do not become a part of the template, and thus will vanish upon redrawing, zooming .... See Examples.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

figAddReservoirs

points triplotadd

figaro par

Examples

```
data(sazava)
accessVar("sazava")

setCex(1.5)
pokeDataset("sazava", overwrite.warn=FALSE)

# Calculate Rayleigh-type fractionation trend
ff<-seq(1,0.1,-0.1) # F, amount of melt left
x<-80*ff^(1.2-1)    # cL for three elements, arbitrary D of 1.2, 2.0 and 1.3
y<-550*ff^(2.0-1)
z<-1000*ff^(1.3-1)
my.trend<-cbind(x,y,z)
colnames(my.trend)<-c("Rb","Sr","Ba")
rownames(my.trend)<-ff

# By default, the overplotted information is added permanently
binary("Rb","Sr",log="xy")
figOverplot(var.name="my.trend",pch="+",col="blue",autoscale=TRUE,type="o",
    labs=rownames(my.trend))
figXlim(c(30,100))

# But this is controlled by the argument just.draw
binary("Rb","Sr",log="xy")
figOverplot(var.name="my.trend",pch="+",col="darkred",autoscale=TRUE,type="o",
    labs=rownames(my.trend),just.draw=TRUE)
# Any function redrawing the plotting window will wipe the added trend out
figXlim(c(30,100))

# Ternary
ternary("10*Rb","2*Sr","Ba/2")
figOverplot(var.name="my.trend",pch="+",col="blue",type="o",
    labs=rownames(my.trend))
```

`figRedraw` *Redrawing/refreshing a Figaro plot*

Description

These functions redraw/refresh a Figaro-compatible plot.

Usage

```
figRedraw(x=x.data, y=y.data, zoom=NULL, bw=FALSE, title=NULL)

refreshFig()
```

Arguments

`x`	vector of x coordinates
`y`	vector of y coordinates
`zoom`	logical; redraw while zooming?
`bw`	logical; should be the output black and white?
`title`	character; optional title for the plotting window.

Warning

Note that all user-defined components added via 'Plot editing: Add' (legend, lines, text, boxes, ...) - will be lost.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

and Vojtech Janousek, <vojtech.janousek@geology.cz>

See Also

figaro

Examples

```
data(sazava)
accessVar("sazava")

binary("SiO2","Na2O+K2O",main="My TAS diagram")
windows()

figRedraw()
```

---

figScale *Plot editing: Scaling text or plotting symbols*

Description

These functions enable changing a size of titles, axis labels or plotting symbols of Figaro-compatible plots. The size is relative to 1 (the original).

Usage

```
figCex(x=NULL,redraw=TRUE)

figCexLab(x=NULL,redraw=TRUE)

figCexMain(x=NULL,redraw=TRUE)

figCexSub(x=NULL,redraw=TRUE)
```

Arguments

x	numeric: scaling factor.
redraw	logical; should be modified Figaro template redrawn?

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

and Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'figaro'

Examples

```
data(sazava)
accessVar("sazava")

binary("SiO2","Na2O+K2O",main="My TAS diagram")
figSub(txt="on Sazava suite")
figCex(2)
figCexMain(1.5)
figCexSub(0.5)
```

---

figUser *Plot editing: User defined parameter*

Description

Enables the power users to modify the plotting parameters directly, one or several at the time.

Usage

```
figUser(expression=NULL,...,redraw=TRUE)
```

Arguments

expression	character; single text string to be evaluated that contains parameters to be changed.
...	(alternative) list of named parameters to be changed.
redraw	logical; should be the modified Figaro template redrawn?

Details

The parameters can be specified at the function call, either as a single expression (text string) or as a list of named parameters in the form `par.name = value` (see examples). If neither is specified, the parameters are chosen by a GUI dialogue. If no parameters are entered from the GUI, they can be chosen from a list (still experimental!)

Several of parameters can be entered simultaneously.

NB that in the single text expression, quotation marks need to be preceded by backslashes as an escape character. In the expression, individual parameters are to be separated by semicolons.

Arguably the most useful parameters are (see par for further possibilities):

main	main title
sub	sub title
xlab	label of x axis
ylab	label of y axis
xlim	limits for the x axis
ylim	limits for the y axis
bg	colour of background
pch	plotting symbols

col	colour of plotting symbols
cex	relative size of plotting symbols
cex.axis	magnification for axis annotation
cex.lab	magnification for x and y labels
cex	relative size of plotting symbols
log	which of the axes is logarithmic? (`""`,`"x"`,`"y"` or `"xy"`)

Menu

Plot editing: User defined parameter

Warning

If requesting a logarithmic plot, do make sure that the axis ranges are all positive. See Examples or invoke menu items '`Plot editing: Scale x axis`' and '`Plot editing: Scale y axis`'.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

and Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

figaro, par, plateUser

Examples

```
data(blatna)
accessVar("blatna")

binary("SiO2","Na2O+K2O",main="My TAS diagram")
figUser(pch="+")

figUser(col="darkblue")

figUser(pch=1,col=2,cex=1.5)

figUser(bg="khaki", cex=1.5) # for camouflage purposes

# For power users
figUser(main="My plot 1", las=3, col.main="blue", cex.main=2.5, cex.lab=1.8, cex.axis=0.75)

# Alternative syntax
figUser("main=\"My plot 2\"; las=2; cex.lab=1.2; cex.axis=1; col.main=\"darkred\"")
```

---

figZoom *Plot editing: Zooming*

Description

These functions zoom in and out Figaro-compatible plots.

Usage

```
figZoom()

figUnzoom()

figXlim(range=NULL)

figYlim(range=NULL)

figFixLim(no.action.warn=TRUE)
```

Arguments

range	numeric: two limits, minimum and maximum, for the given axis.
no.action.warn	logical: should be a warning shown if there is no action needed?

Details

'figZoom' zooms the specified rectangular area (click bottom left and then top right corner) in a new window. The zoomed area is highlighted in the old window.

'figUnzoom' closes the new window with blown up portion of the plotting window and returns to the original onr.

'figXlim' and 'figYlim' allow to change the plotting limits (as a list of two components, separated by commas).

'figFixLim' extends the scales of both axes of a binary plot automatically if necessary to accommodate all the data points.

Warning

If requesting a logarithmic plot, make sure that the axis ranges are positive.

Author(s)

Colin M. Farrow, <colinfarrow537@gmail.com>

and Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'figaro'

Examples

```
data(sazava)
accessVar("sazava")

binary("SiO2","Na2O+K2O",main="My TAS diagram")
figCex(1.5)
figXlim(c(45,65))
figYlim(c(0,10))

## Not run:
    figFixLim(no.action.warn=TRUE)
    figFixLim(no.action.warn=TRUE)
```

```
        figFixLim(no.action.warn=FALSE)

        figZoom()

## End(Not run)
```

`filledContourFig` *Filled contours plot*

Description

Generates a frequency plot on the basis of the most recently plotted Figaro template.

Usage

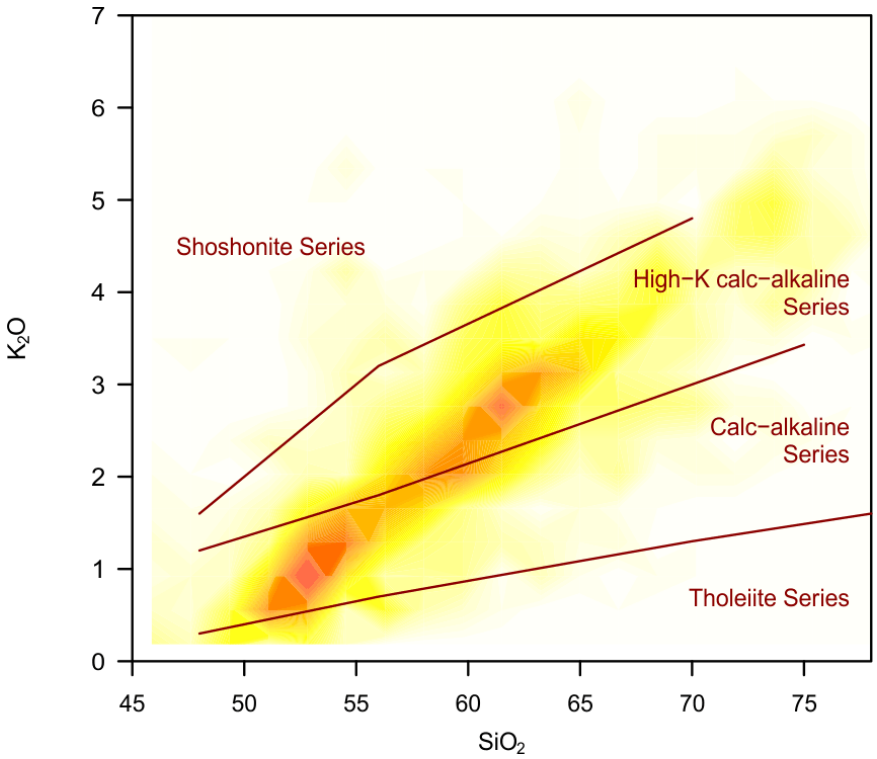
```
filledContourFig(palette="heat.colors",grid.density=NULL,overplot=FALSE,
xlab=sheet$demo$call$xlab,ylab=sheet$demo$call$ylab,
xlim=sheet$demo$call$xlim,ylim=sheet$demo$call$ylim,
annotate.fields=FALSE,...)
```

Arguments

`palette`	character; name of the palette
`grid.density`	numeric; density of the grid along the x and y axes
`overplot`	logical; should be the diagram overplotted by the original dataset?
`xlab`	character vector; label for the x axis
`ylab`	character vector; label for the y axis
`xlim`	limits for the x axis
`ylim`	limits for the y axis
`annotate.fields`	logical; should be the plotted fields labeled by their names?
`...`	additional plotting parameters

Details

This is a somewhat modified version of the R function '`filled.contour`' that produces a frequency plot on the basis of a Figaro template and superimposes, if desired, selected data points.



The user can specify how many intervals should be each of the axes split into ('`grid.density`'). This corresponds to a density of the grid, in which are the individual points classified into. A bit of experimenting with this values is usually needed. Then a colour scheme ('`palette`') can be chosen. After the frequency plot is generated, selected analyses can be plotted on the top (`'overplot'`).

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'addContours' 'figaro' 'selectPalette'

Examples

```
data(atacazo)
accessVar("atacazo")

binary("SiO2","Na2O+K2O",main="My TAS diagram")
filledContourFig(palette="heat.colors",grid.density=15)

# User-defined palette
binary("SiO2","Na2O+K2O",main="My TAS diagram")
my.palette<-colorRampPalette(c("darkred", "red","pink", "white"),space = "rgb")
filledContourFig(palette="my.palette",grid.density=15,xlim=c(58,70),ylim=c(0,10))
```

formula2vector *Counting individual atoms in a formula unit*

Description

Function counting number of individual atoms in a formula given by a simple text string.

Usage

```
formula2vector(formula)
```

Arguments

formula	character; a mineral formula

Details

This function converts a mineral formula given as a single character string - like.g. 'KAlSi3O8' for K-feldspar - to a vector with all atoms (per formula unit) counted. Elements can be repeated, trivalent Fe has to be denoted as 'FeIII'. Brackets can be used, like '(PO4)2'.

Value

A named numeric vector with numbers of individual atoms.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

idealMineralCompositions plComp olComp minComp

Examples

```
formula2vector("NaAlSi3O8")         # Number of apfu per albite formula

formula2vector("6NaAlSiO4·Na2CO3")  # Cancrinite

formula2vector("CaMg(CO3)2")        # Dolomite

formula2vector("Ca3FeIII2Si3O12")   # Andradite
```

garnet *Selected garnet analyses*

Description

This data set gives the selected chemical analyses of garnet from *Deer et al. (2013)*.

Usage

```
data(garnet)
```

Format

A data frame containing 6 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 20 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(garnet)
accessVar("garnet")
print(WR)
```

gcdOptions *GCDkit options*

Description

A graphical user interface (GUI, programmed in Tcl/Tk) for setting the main options controlling the behaviour of the GCDkit.

Usage

```
gcdOptions(permanent.only=FALSE)
```

Arguments

permanent.only	logical; should be shown exclusively the option that can be set permanently?

Details

The settings are stored permanently in the file 'gcdkit.xxx' residing in the main GCDkit directory. They are loaded upon start up. If is missing or damaged, this file is created anew based on the default values.

The panel connected to the function 'gcdOptions' serves to change several parameters. Most of them are passed to a list accessible in a way similar to the standard R options. See the corresponding manual page for details and Examples for their implementation. Only a few are stored in dedicated variables (see below).

Firstly, the default working directory can be set (and stored in the global variable data.dir).

Secondly, the default Graphical User Interface (GUI) including the menu system can be specified using the option gcd.menus. It can attain one of the three values: "","win","tcltk".

If GCDkit is run under Windows OS using RGui (standard behaviour), a default value of the gcd.menus parameter is set to "win" automatically.

Otherwise (any other operating system, Windows in batch mode using the RTerm window), the default value is "" unless it has been modified previously in the configuration file. In these cases, "tcltk" would be the correct setting.

From within GCDkit under any operating system, Tcl/Tk interface can be (re)started anytime using the menuet command.

The parameter 'Minimize output on screen?' is linked to the option gcd.shut.up. It controls excessive output to the Console window. Its default value is FALSE, meaning that detailed information is to be printed. This, however, may become not viable on slower systems and/or for extensive data sets.

The preferred precision of the numeric values that need to be rounded off are controlled by the parameter 'Precision of results' (option gcd.digits).

Using the parameter 'Plotting symbols magnification', linked to the option gcd.cex, one can define a factor, by which are multiplied the plotting sizes defined for individual analyses upon startup and stored in the variable 'labels[,"Size"]'. Please note that this is effective for the next plot if the GUI frontend is used to set this parameter, otherwise it will work for data files loaded from now on.

In this way, the magnification is maintained proportional to the original sizes. If uniform plotting symbols sizes are desired, one should use the function setCex invoked from the menu

Plot settings|Set uniform symbol size.

The parameter 'Annotate fields in discrimination plots?' toggles the labeling of the fields on and off, typically for classification or geotectonic diagrams. It is stored in a logical variable gcd.plot.text, whose default is TRUE. The language for the field annotations can be selected using the list box connected to the option gcd.language.

The next possibility is to alter the colours used, e.g, for texts or field boundaries on diagrams. There are in total three colours stored in the list plt.col. Alternatively, all the plots can be set to black and white (check box 'Set to BW?' linked to the option 'gcd.plot.bw'), excluding the data points. The default is FALSE (i.e. colour plotting).

The parameter 'Identify points?' toggles on and off the identification/labelling of individual data points on plots. In general, the identification can be either interactive (option
gcd.ident.each = TRUE) or all the points can be labeled automatically as soon as the plotting is finished (option
gcd.ident.each = FALSE). In the former case, the user may click the left mouse button near the points to be identified, pressing the right mouse button when finished.

The option gcd.ident determines whether identification should take place at all (the default value is zero, which means no identification). If the identification is on, the option gcd.ident attains

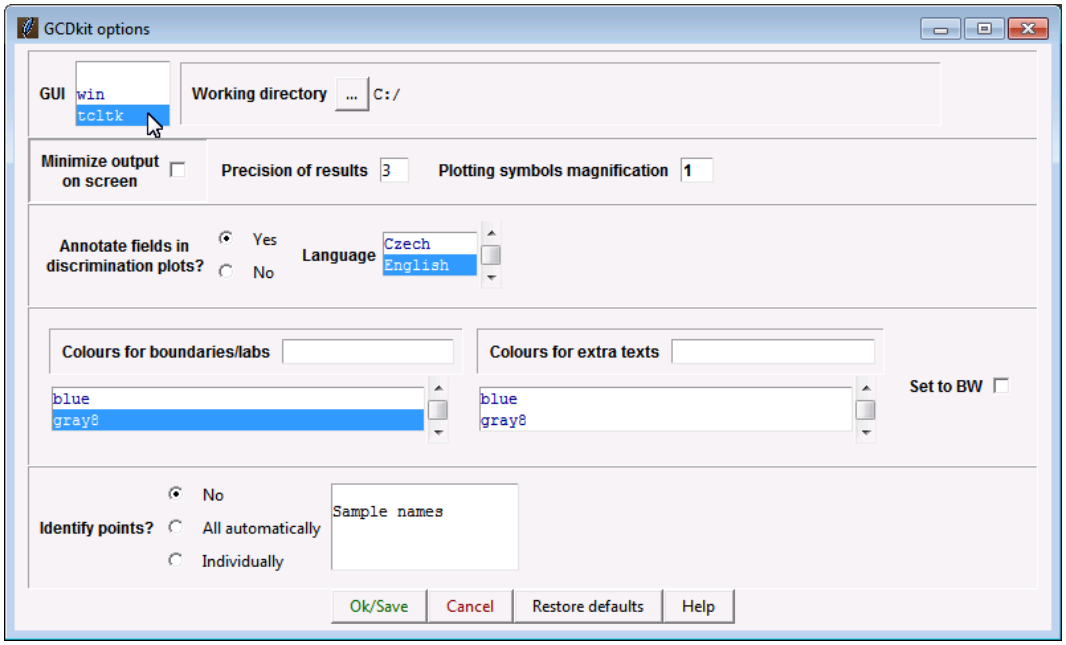
either 1 (identification by sample name), or the sequential number of the column in the data frame 'labels' increased by one (identification by a label).

The identification by sample name for a current plot can be invoked also from the menu 'Plot editing|Identify points'. There can be also chosen alternative means of points identification ('Plot editing|Highlight multiple points').

Value

Sets the following options:

gcd.plot.text	logical; should be fields on classification diagrams labeled by their names?
gcd.language	language for these labels.
gcd.plot.bw	logical; if TRUE, plots are produced as black and white.
gcd.shut.up	logical; determines whether extensive textual output is to be printed.
gcd.ident	numeric; if zero, no identification takes place after plotting each diagram. If greater than zero, indicates the variable used to identify individual data points. See Details.
gcd.ident.each	logical; are the data points to be identified individually?
gcd.digits	preferred number of digits for rounding off the numeric values.
gcd.cex	a factor by which are multiplied all symbol sizes previously defined.



Remaining options changed by GCDkit which cannot be altered via the GUI, though:

prompt	"GCDkit-> "
windowsBuffered	FALSE
locatorBell	FALSE
scipen	20
max.print	99999999

If necessary they can be set directly in the file 'gcdkit.xxx'.

Apart from that the GUI panel sets the variables data.dir (default data directory) and plt.col (colours for Figaro-compatible plots).

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

options identify ID figaro setCex menuet

Examples

```
bak <- options()                # backup the current options
options("gcd.ident"=1)          # identify by sample names
options("gcd.ident.each"=FALSE) # to label by sample names automatically,
                                # i.e. without the user interference

plotDiagram("TAS",FALSE,FALSE)
options("gcd.ident"=0)          # to turn off the identification completely
options("gcd.plot.bw"=TRUE)     # to set the diagram to black and white
plotDiagram("TAS",FALSE,FALSE)

options("gcd.cex"=2)            # make the plotting symbols double as big
                                # (effective for the data files loaded from now on;
                                # for immediate result use the GUI front end)

getOption("gcd.plot.bw")        # printing the current value of the given option
options(bak)                    # restore the previous options
```

---

graphicsOff *Close all graphic windows*

Description

Closes all graphic windows.

Usage

```
graphicsOff()
```

Arguments

None.

Details

Sometimes, the R system may become slow, failing to redraw graphical windows if too many of them are being open. It is always a good idea to close the unnecessary ones, for instance using this function.

See Also

'dev.off'

groupsByLabel *Groups by label*

Description

Grouping the data according to the levels of a single label.

Usage

```
groupsByLabel(lab=NULL)
```

Arguments

lab	name or sequence number of the label

Details

Sets the groups on the selected column within the data frame 'labels'. If not specified at the function call, the appropriate label is selected by the function 'selectColumnLabel'.

Value

groups	character vector: the grouping information
grouping	the sequence number of the column in the data frame 'labels' used for grouping

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(sazava)
accessVar("sazava")
groupsByLabel("Intrusion")
```

HTMLFormula *Export structural formulae to HTML*

Description

These functions output structural formulae of specified mineral(s) into HTML. This format is useful for publishing on the World Wide Web or exporting (e.g., one can select the whole WWW page, copy and paste it to a into a word processor).

Usage

```
HTMLFormula(mineral,digits=3,browse=TRUE)
HTMLFormula.all(minerals=NULL,digits=3,browse=TRUE)
HTMLFormula.Jupyter(mineral=NULL,digits=3)
```

Arguments

mineral	character of the length 1; name of mineral class for output
minerals	character; names of mineral classes for output
digits	numeric: precision of the output
browse	logical: should be the HTML output displayed in browser?

Details

'HTMLFormula' returns the formulae for all the analyses of the given mineral. The browser with the resulting web page starts automatically.

'HTMLFormula.all' is the front end to 'HTMLFormula', enabling the user to choose one or more minerals for the output.

'HTMLFormula.Jupyter' is an interface function to 'HTMLFormula' to produce a Jupyter-compatible HTML output.

Atoms with concentrations lower than 0.5 times the required precision ('digits') are omitted.

Value

Functions return (invisibly) a character matrix with html code.

Warning

All these functions require the 'R2HTML' library; the function HTMLFormula.Jupyter also the library 'IRdisplay'. These must be downloaded from the CRAN and properly installed. Their presence is checked before the code is executed.

Warning

All these functions require the 'R2HTML' library. It must be downloaded from the CRAN and properly installed. Their presence is checked before the code is executed.

Additionally, function HTMLFormula.Jupyter requires a correctly configured IR kernel connection, including the 'IRdisplay' library.

Remark

The function HTMLFormulaC contains code identical to HTMLFormula, compiled for speed.

Author(s)

The R2HTML package was written by Eric Lecoutre.

The IRdisplay package was written by Thomas Kluyver, Philipp Angerer and Jan Schulz.

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class

minAssign

minCheckValency

minFormula

minAllocateAtoms

minValues

minEndMembers

minMain

Examples

```
data(combined)
accessVar("combined")

HTMLFormula("garnet")
```

ID *Sample identification*

Description

Identification/labelling of individual data points on plots.

Usage

```
ID(x, y, labs=getOption("gcd.ident"), offset=0.4,
    col="gray30", cex=1)
```

Arguments

x, y	vector with x-y coordinates of the data points
labs	text to label individual data points, see details
offset	distance (in char widths) between label and identified points.
col	colour of the text
cex	its size

Details

In GCDkit, the option 'ident' determines whether the user wishes to identify data points on binary and ternary plots. The default is zero, which means no identification.

If 'ident' differs from zero, internal function 'ID' can be invoked. Its parameter labs is either a single number, or character vector.

In the former case, the variable 'labs' contains either 1 (identification by sample name), or the sequential number of the column in the data frame 'labels' increased by one (identification by a user- defined label).

Alternatively, a character vector labs can be used to specify the text directly.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

identify gcdOptions options

Examples

```
getOption("gcd.ident")  # yields the current value of the given option
```

`info` *Info on datafile*

Description

Prints information about the current dataset (and its selected subset, if applicable).

Usage

```
info()
```

Details

This function prints comprehensive information about the current dataset. For each of the labels, individual levels and their frequencies are given. The number of numeric columns is printed, and for each of the variables number of available values. Moreover, the information concerning the total number of samples, the names of the samples in the selected subset (or all samples if none is defined) and the current grouping are shown.

Value

None

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

`joinGroups` *Merge groups*

Description

Enables merging several groups into a single one.

Usage

```
joinGroups()
```

Arguments

None.

Details

This function is the most useful to merge several groups, defined e.g. on the basis of a classification plot. A simple spreadsheet is invoked with two columns, the first (`'Old'`) containing the old levels of groups and the second, `'New'`, which can be edited. Finally, groups with identical names will be merged into a single one.

Optionally, the vector containing the information on the current groups can be appended to the data frame '`labels`'.

Value

`groups`	character vector: the grouping information
`grouping`	Sequential number of the column with grouping information in `labels` (if appended) or simply set to -100.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

---

loadData *Loading data into GCDkit*

Description

Loads data from a file (or, alternatively, a clipboard) into GCDkit. The files may contain plain text, or, if library RODBC (has been installed, can be in the dBase III/IV (*.dbf), Excel (*.xls), Access (*.mdb), PetroGraph (*.peg), IgPet or NewPet (*.roc) formats.

Usage

```
loadData(filename=NULL,separators = c("\t", ",", ";"," "),
na.strings = c("NA","-","bd", "b.d.", "bdl", "b.d.l.", "N.A.","n.d."),
clipboard = FALSE, merging = FALSE);

loadDataOdbc(filename=NULL,na.strings=c("NA","-", "bd",
"b.d.", "bdl", "b.d.l.", "N.A.","n.d."),merging=FALSE,
ODBC.choose=TRUE)
```

Arguments

`filename`	fully qualified name of the file to be loaded, including suffix.
`separators`	strings that should be tested as prospective delimiters separating individual items in the data file.
`na.strings`	strings that will be interpreted, together with empty items, zeros and negative numbers, as missing values (`NA`).
`clipboard`	logical; is clipboard to be read instead of a file?
`merging`	logical; is the function invoked during merging of two data files?
`ODBC.choose`	logical; if TRUE, ODBC channel can be chosen interactively.

Details

If library RODBC is available, the functions attempt to establish an ODBC connection to the selected file, and open it as dBase III/IV (*.dbf), Excel (*.xls/*.xlsx) or Access (*.mdb) format. The DBF files are used to store data by other popular geochemical packages, such as IgPet *(Carr, 1995)* or MinPet *(Richard, 1995)*.

Another format that can be imported if RODBC is available, is *.csv. It is employed by geochemical database systems such as GEOROC (https://georoc.mpch-mainz.gwdg.de//georoc/) and PETDB (https://search.earthchem.org).

The import filter for the *.csv files has been tailored to keep the structure of these databases in mind.

The package PetroGraph *(Petrelli et al. 2005)* saves data into *.peg files that are also, in principle, *.csv files compatible with the GCDkit.

Data files *.roc are yet another variant of *.csv files, used by NewPet *(Clarke et al. 1994)*. This is not to be confused with the *.roc format designed for IgPet *(Carr, 1995)*. This is a text file with a quite complex structure, whose import is still largely experimental. DBF files are to be preferred for this purpose.

If not successful, the function '`loadData`' assumes that it is dealing with a simple text file.

On the other hand '`loadDataOdbc`' allows an ODBC channel to be specified interactively if '`ODBC.choose=TRUE`'.

Plain text files can be delimited by tabs, commas or semicolons (the delimiter is recognized automatically). Alternative separators list can be specified by the optional '`separators`' parameter. The Windows clipboard is just taken as a special kind of a tab-delimited text file.

In the text file, the first line contains names for the data columns (except for the first one that is automatically assumed to contain the sample names); hence the first line may (or may not) have one item less than the following ones. The data rows start with sample name and do not have to be all of the same length (the rest of the row is filled by '`NA`' automatically).

Missing values ('`NA`') are allowed anywhere in the data file (naturally apart from sample and column names); any of `'NA','N.A.','-','b.d.','bd','b.d.l.','bdl'` or `'n.d.'` are also treated as such, as specified by the parameter `na.strings`.

While loading, the values '`#WHATEVER!`' (Excel error messages) are also replaced by '`NA`' automatically.

Please note that the function '`loadDataOdbc`', due to the current limitations of the RODBC package, cannot handle correctly columns of mixed numeric and textual data. In such a column all textual information is converted to `'NA'` and this unfortunately concerns the sample names as well. If encountering any problems, please use import from text file or via clipboard, which are much more robust.

The negative numbers and values '`< x`' (used by some authors to indicate items below detection limit) can be either replaced by their half (i.e. half of the detection limit) or '`NA`'. User is prompted which of these options he prefers.

Alternatively, the negative values can be viewed either as missing ('`NA`') or can be imported, as may be desirable for instance for stable isotope data in the delta notation.

Decimal commas, if present in text file, are converted to decimal points.

The data files can be practically freeform, i.e. no specified oxides/elements are required and no exact order of these is to be adhered to. Analyses can contain as many numeric columns as necessary, the names of oxides and trace elements are self-explanatory (e.g. `"SiO2","Fe2O3","Rb","Nd"`.

In the text files (or if pasting from clipboard), any line starting with the hash symbol (`'#'`) is ignored and can be used to introduce comments or to prevent the given analysis from loading temporarily.

Note that names of variables are case sensitive in R. However, any of the fully upper case names of the oxides/elements that appear in the following list are translated automatically to the appropriate capitalization:

```
SiO2, TiO2, Al2O3, Fe2O3, FeO, MnO, MgO, CaO, Na2O, FeOt, Fe2O3t,

Li2O, mg#, Ac, Ag, Al, As, At, Au, Ba, Be, Bi,

Br, Ca, Cd, Ce, Cl, Co, Cr, Cs, Cu, Dy, Er, Eu,

Fe, Ga, Gd, Ge, Hf, Hg, Ho, In, Ir, La, Li, Lu,

Mg, Mn, Mo, Na, Nb, Nd, Ne, Ni, Np, Os, Pa, Pb,

Pd, Pm, Pr, Pt, Pu, Rb, Re, Rh, Ru, S, Sb, Sc,

Se, Si, Sm, Sn, Sr, Ta, Tb, Te, Th, Ti, Tl, Tm,

Yb, Zn, Zr.
```

Total iron, if given, should be expressed either as ferrous oxide ('FeOt', 'FeOT', 'FeOtot', 'FeOTOT' or 'FeO*') or ferric oxide ('Fe2O3t', 'Fe2O3T', 'Fe2O3tot', 'Fe2O3TOT' or 'Fe2O3*').

Structurally bound water can be named 'H2O.PLUS', 'H2O+', 'H2OPLUS', 'H2OP' or 'H2O_PLUS'.

Upon loading, all the completely empty columns are removed first. Any non-numeric items found in a data column with one of the names listed in the above dictionary are assumed to be typos and replaced by 'NA', after a warning appears. At the next stage all fully numeric data columns are stored in a numeric data matrix 'WR'.

For any missing major- and minor-element data (SiO2, TiO2, Al2O3, Fe2O3, FeO, MnO, MgO, CaO, Na2O, K2O, H2O.PLUS, CO2, P2O5, F, S), an empty (NA) column is created automatically.

The remaining, that is all at least partly textual data columns are transferred to the data frame 'labels'. To this are also attached a column whose name starts with 'Symbol' (if any) that is taken as containing plotting symbols and a column whose name is 'Colour' or 'Color'(if any, capitalization does not matter) that may contain plotting colours specification. The relative size of the individual plotting symbols may be specified in a column named 'Size' or 'cex' that is also to be attached to the 'labels'.

The plotting symbols can be given either by their code (see showSymbols) or directly as strings of single characters.

The colours can be specified as codes (1-49) or English names (see showColours or type 'colours()' into the Console window).

If specifications of the plotting symbols and colours are missing completely, and at least one non-numeric variable is present, the user is prompted whether he does not want to have the symbols and colours assigned automatically, from 1 to *n*, according to the levels of the selected label. Otherwise default symbols (empty black circles) are used.

The default grouping is set on the basis of plotting symbols '(labels$Symbol)' or the data column used to autoassign the plotting symbols and colours.

Lastly, a backup copy of the data is stored in the list 'WRCube' using the function 'pokeDataset'. It is stored either under the name of the file, or, if it already exists, under the file name with a time stamp attached.

Value

WR	numeric matrix: all numeric data
labels	data frame: all at least partly character fields; labels$Symbol contains plotting symbols and labels$Colour the plotting colours

The function prints a short summary about the loaded file. It also loads and executes the Plugins, i.e. all the R code (*.r) that is currently stored in the subdirectory '\Plugin'. Finally, the system performs some recalculations (calling 'Gcdkit.r').

Warning

The RODBC package (and thus also the import from CSV, Excel (*.xls/*.xlsx), Access (*.mdb) or dBase III/IV (*.dbf) files), is not available on 64-bit systems!

Note

In order to ensure the database functionality, duplicated column (variable) names are not allowed. This concerns, to a large extent, also the sample names. The only exception are CSV files - if duplicated samples are found, sequence numbers are assigned instead.

All completely empty rows and columns in both labels and numeric data are ignored.

Author(s)

The RODBC package was written by Brian Ripley.

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Carr M (1995) Program IgPet. Terra Softa, Somerset, New Jersey, U.S.A.

Clarke D, Mengel F, Coish RA, Kosinowski MHF(1994) NewPet for DOS, version 94.01.07. Department of Earth Sciences, Memorial University of Newfoundland, Canada.

Petrelli M, Poli G, Perugini D, Peccerillo A (2005) PetroGraph: A new software to visualize, model, and present geochemical data in igneous petrology. Geochemistry Geophysics Geosystems 6: 1-15

Richard LR (1995) MinPet: Mineralogical and Petrological Data Processing System, Version 2.02. MinPet Geological Software, Quebec, Canada.

See Also

'saveData' 'mergeData' 'pokeDataset' 'showColours' 'showSymbols' 'read.table' 'getwd' 'setwd'

Examples

```
# Sets the working path and loads the 'sazava' test data set
setwd(paste(gcdx.dir,"Test_data",sep="/"))
loadData("sazava.data")
```

mergeData *Appending data to a current data set*

Description

These functions append new data to the analyses currently stored in the memory of the GCDkit.

Usage

```
mergeDataRows(which.file=NULL)
mergeDataCols(which.file=NULL,all.rows=NULL)
```

Arguments

which.file	optional filename of th file to be appended.
all.rows	logical; should be all samples preserved, even those missing in one of the datasets?

Details

The function 'mergeDataRows' appends new samples (i.e. new rows). The structures of both datafiles are, as much as possible, matched against each other, and, if necessary, new empty columns are introduced to the original data file, if they are missing. If any duplicated sample names are found, they are replaced by sequence numbers and a new column 'old.ID' is appended to the labels. Also appended is a column named 'file' containing the name of the file the particular sample originated from.

'mergeDataCols' adds new data (i.e. new data columns) to the samples stored in the memory. If desired ('all.rows' is 'TRUE'), included are also samples that occur solely in one of the files.

For the guidelines on correct formatting of the data files see loadData.

Value

WR	numeric matrix: all numeric data
labels	data frame: all at least partly character fields; labels$Symbol contains plotting symbols and labels$Colour the plotting colours

The function prints a short summary about the loaded file.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'loadData' 'saveData' 'merge'

micas *Selected analyses of micas*

Description

This data set gives the selected chemical analyses of micas from *Deer et al. (2013)*.

Usage

```
data(micas)
```

Format

A data frame containing 8 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 179 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(micas)
accessVar("micas")
print(WR)
```

minAllocateAtoms *Allocate atoms to crystallographic sites*

Description

Allocates the recalculated atoms per formula unit to crystallographic sites of the given mineral(s).

Usage

```
minAllocateAtoms(object, silent=FALSE)
minAllocateAtomsC(object, silent=FALSE)
minAllocateAtomsAll()
```

Arguments

object	of the class "mineral"; data and recalculation options for a single mineral
silent	logical; warn if some atoms are not allocated in full

Details

The sites are filled by individual atoms in the order specified, from left to right, in the slot 'sites'. If some of the atoms may be present in two sites, the slot 'site.sums' must give the required sum for the first of them. When a site is filled (the sum in 'site.sums' reached or exceeded), the excess of the given atom is passed to the next position available.

In the following hypothetical example:

```
sites=list(Z=c("Si","Al"),X=c("Mg","Fe"),Y=c("Ti","Al")),
site.sums=c(6,NA,NA),
```

the site 'Z' is filled by all Si and part of Al (6 - Si). The excess Al will be allocated to the site 'Y'. In other words:

```
Al_Z = 6 - Si_Z; Al_Y = Al - Al_Z
```

In this case, no sums had to be defined for sites 'X' and 'Y', hence the value 'NA' standing for 'not available'.

If desired, a special 'atom name' for vacancy (Vc) can be used if the site is to be filled up by vacancies to the sum specified.

'minAllocateAtoms' and 'minAllocateAtomsAll' perform the allocations for all the analyses of the given mineral or the whole dataset respectively.

Value

'minAllocateAtoms' returns a numeric matrix with the atom allocation for the given analyses and mineral class. The individual columns in the matrix are named by atom name, an underscore and the site name as specified in the slot 'sites'.

'minAllocateAtomsAll' returns a list with items for each of the mineral species. Therein, it allocates the atoms, for each mineral separately, to the slot 'formula'.

Remark

The function minAllocateAtomsC contains code identical to minAllocateAtoms, compiled for speed.

Warning

Make sure that the atoms than can overflow into one of the following sites are always listed last. If not, some of the elements/atoms may remain unallocated.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class

minAssign

minCheckValency

minFormula

minValues

minEndMembers

minClassify

minMain

Examples

```
data(combined)
accessVar(combined)
out<-minAllocateAtoms(min.data$feldspar, silent=FALSE)
print(out)

out<-minAllocateAtomsAll()
print(out)
```

`minAssign` *Assign analyses to mineral species*

Description

This function splits the input data matrix (read from datafile or clipboard) into a list `'min.data'` that contains named component(s) for each of the mineral species (class) present.

Usage

```
minAssign(matrix=WR)
minAssignC(matrix=WR)
```

Arguments

`matrix`	numeric matrix with the data to be classified

Details

The mineral class(es) for individual analyses can be specified already in the data file, in a column named `'Mineral'`. This column can contain a mixture of the full names and/or abbreviations as specified, for each mineral class, in the database slots `'full'` and `'abbreviated'`. The abbreviations include the standard ones from *Kretz (1983)*, *Whitney & Evans (2010)* or *Warr (2021)* as well as those used by other packages, e.g. MinPet *(Richard 1995)*, ThermoCalc *(Holland & Powell 1998)* and PET *(Dachs 1998, 2004)*. For instance, the garnet entry in the database may look like:

```
abbreviated=c("Grt","Alm","Prp","Grs","Sps","Adr","Uv","GAR","g"),
full=c("garnet","almandine","pyrope","grossular","spessartine")
```

If no such a column is present or if it contains some unrecognized names, the desired mineral is to be chosen from a drop-down list of the mineral classes available in the database.

All the mineral data are finally assigned to a list `'min.data'` with named components for each of the mineral species (e.g. `'min.data$garnet'`). All numeric data are stored in a slot `'rough'` (`'min.data$garnet@rough'`), all at least partly textual data in the slot `'labels'` as are the plotting attributes (plotting symbols, their colours and relative size).

Value

Assigns the imported numeric data, for each mineral class separately, into a list `'min.data'`.

Remark

The function `minAssignC` contains code identical to `minAssign`, compiled for speed.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Dachs E (1998) PET: petrological elementary tools for Mathematica. Comp Geosci 24: 219-235 doi: 10.1016/S00983004(97)001416doi: 10.1016/S0098-3004(97)00141-6

Dachs E (2004) PET: Petrological Elementary Tools for Mathematica: an update. Comp Geosci 30: 173-182 doi: 10.1016/j.cageo.2003.09.007doi: 10.1016/j.cageo.2003.09.007

Holland TJB, Powell R (1998) An internally consistent thermodynamic data set for phases of petrological interest. J Metamorph Geol 16: 309-343 doi: 10.1111/j.15251314.1998.00140.xdoi: 10.1111/j.1525-1314.1998.00140.x

Kretz R (1983) Symbols for rock-forming minerals. Am Min 68: 277-279 doi: 10.1111/j.1525-1314.1998.00140.xdoi: 10.1111/j.1525-1314.1998.00140.x

Richard LR (1995) MinPet: Mineralogical and Petrological Data Processing System, Version 2.02. MinPet Geological Software, Québec, Canada.

Warr, L.N. (2021) IMA-CNMNC approved mineral symbols. Mineralogical Magazine, 85, 291-320. doi: 10.1180/mgm.2021.43 doi: 10.1180/mgm.2021.43doi: 10.1180/mgm.2021.43

Whitney DL & Evans BW (2010) Abbreviations for names of rock-forming minerals. Am Min 95: 185-187 doi: 10.2138/am.2010.3371doi: 10.2138/am.2010.3371

See Also

`mineral-class`

`minCheckValency`

`minFormula`

`minAllocateAtoms`

`minValues`

`minEndMembers`

`minClassify`

`minMain`

Examples

```
names(getClass("mineral")@subclasses)                 # list of mineral classes

sapply("garnet",.getClassPrototype,slot="full")       # all full names

sapply("garnet",.getClassPrototype,slot="abbreviated") # all abbreviations
```

`minCheckValency` *Check the atoms in imported data for appropriate valency*

Description

These auxiliary functions check whether the valency of atoms in oxides in the imported data file are compatible with the mineral's formula.

Usage

```
minCheckValency(object)
minCheckValencyC(object)
minCheckValencyAll()
```

Arguments

`object`	of the class "mineral"; data and recalculation options for a single mineral

Details

The function `'minCheckValency'` works in two steps:

(1) For most of the oxides, the algorithm checks whether the input data correspond to the usual valency states of the individual atoms. If not, the function recasts the imported oxides to those occurring the most commonly in the nature.

(2) If either of FeO or Fe_2O_3 has not been determined, and no Fe^{II}/Fe^{III} estimation is desired, the function cross-checks the iron valency in the data file and in the mineral formula. The latter is taken from the slot `'atoms'`. Again, in case of a mismatch, the imported oxides are recast to either FeO or Fe_2O_3, as appropriate.

`'minCheckValencyAll'` is a wrapper function that performs the check for the whole dataset (i.e., all minerals).

Value

`'minCheckValency'` and `'minCheckValencyAll'` assign all the checked/amended oxide data to the slot `'rough'` of the object(s) defined for the given mineral(s). In addition the functions return a numeric matrix/list of numeric matrices containing the amended chemical data.

Remark

The function `minCheckValencyC` contains code identical to `minCheckValency`, compiled for speed.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class

minAssign

minAllocateAtoms

minValues

minEndMembers

minClassify

minMain

Examples

```
minCheckValencyAll()
```

`minClassify` *Classify the analyses of the given mineral*

Description

Front-end functions to the classification engine built into *GCDkit.Mineral*. The classification schemes for individual mineral classes can combine binary or ternary plots and pre-defined simple R scripts into, in essence, hierarchical decision trees.

Usage

```
minClassify(mineral=NULL,warning=TRUE)
minClassifyPlot(mineral=NULL,warning=TRUE,all=TRUE,new=TRUE)
```

Arguments

`mineral`	character; which mineral class should be classified?
`warning`	logical; warn when the desired mineral is not present in the data file?
`all`	logical; should be plotted all diagrams found for the given mineral?
`new`	logical; should be a new plotting window opened?

Details

The function `'minClassify'` classifies all the analyses belonging to the given mineral class specified as an argument to the function call. If none is given, the user is prompted to select one from the list of mineral species present in the current data file.

The function calls the classification engine that makes possible a classification of the given mineral species by a pre-defined hierarchical sequence of plots (binary and/or ternary), combined, if necessary, with R language scripts.

`'minClassifyAll'` is front-end to `'minClassify'` . The user can select a mineral species and then one or more classification plots applicable to the given mineral class. If a single graph is desired, the compatibility with Figaro is maintained. This means that the plot can be further retouched using the tools available under the menu `'Plot editing'`. Otherwise a plate of diagrams is produced.

Value

results	character vector with the classification output.

Author(s)

Classification algorithm and the templates have been written by Vojtěch Erban, <erban@sopky.cz>

The wrapper functions added Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class
minAssign
minAllocateAtoms
minValues
minEndMembers
minMain
figaro
Plate

Examples

```
data(combined)
accessVar(combined)

minClassify("clinopyroxene")

minClassifyPlot("clinopyroxene")
```

---

minComp *Computing chemical compositions from mineral formulae*

Description

These functions calculate chemical compositions (in wt.% of oxides) based on mineral's formula(e).

Usage

```
minComp(n.at,oxides=major)
plComp(An=0)
olComp(Fo=1)
idealMineralCompositions(my.mins=NULL)
```

Arguments

n.at	numeric: named vector with numbers of individual atoms
oxides	numeric: names of oxides to be returned
An	numeric: anorthite mass proportion(s), 0-1
Fo	numeric: forsterite mass proportion(s), 0-1
my.mins	character: vector with names of desired minerals

Details

The function minComp takes two arguments. The first is a named vector n.at containing numbers of atoms per formula unit, such as that produced by the function formula2vector. The second contains names of the desired oxide names.

NB that trivalent iron in the formula is to be denoted as 'FeIII'.

The functions plComp and olComp are front ends calculating the chemical compositions of plagioclase or olivine for the respective molar proportions of anorthite (An) and forsterite (Fo) endmembers (0 to 1).

The function idealMineralCompositions calculates chemical compositions (in wt. % of oxides) of the selected rock-forming minerals. If none are specified upon the function's call, they can be selected from a list (GUI). Their ideal mineral formulae are based on (slightly modified) database of *Le Maitre (1982, Appendix 12)*. This table with alphabetic list of ideal mineral molecules is stored, together with originally calculated molecular weights, in the file 'mineral_formulae.data'.

Value

A numeric matrix with calculated major-element compositions.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Le Maitre RW (1982) Numerical Petrology. Developments in Petrology 8. Elsevier, Amsterdam, pp 1-281

See Also

formula2vector

Examples

```
# Albite
z<-formula2vector("NaAlSi3O8")      # Number of individual atoms pfu
minComp(z,c("SiO2","Al2O3","Na2O")) # chemical composition in wt. %

# Magnetite
z<-formula2vector("FeFeIII2O4")
minComp(z) # chemical composition in wt. %

# Plagioclase of given An (0-1)
plComp(0.5)
plComp(seq(0,1,0.1))

# Olivine of given Fo 0-1
olComp(0.25)
olComp(seq(0,1,0.1))

# Andradite
idealMineralCompositions("Andradite")

# Cancrinite
```

```
idealMineralCompositions("Cancrinite")

# Assorted minerals from Le Maitre's database
which.mins<-c("Fluorite","Halite","Magnetite","Quartz","Orthoclase",
    "Albite","Anorthite","Andalusite")
out<-idealMineralCompositions(which.mins)
out<-data.frame(out)
print(out[order(out$SiO2),])    # Sort the output by increasing silica
```

---

minDat *Mineral data (mindat.org)*

Description

Front-end to the mindat.org database.

Usage

```
minDat()
```

Arguments

None.

Details

This function opens a web browser and simply connects to the database of mindat.org. The site contains a searchable database with detailed information on minerals and the localities they are found at.

Value

None.

Author(s)

The database is run by Hudson Institute of Mineralogy, <info@hudsonmineralogy.org>

---

minEndMembers *End-members*

Description

Recasts the analyses into end-member components specified by formulae or R scripts.

Usage

```
minEndMembers(object)
minEndMembersC(object)
minEndMembersAll()
```

Arguments

object	of the class "mineral"; data and recalculation options for a single mineral

Details

The end-member names and definitions can be given in the slots named 'end.member.names' and 'end.member.formulae' stored in the default database, similarly to the case of feldspar:

```
end.member.formulae=c("Na/(Na+Ca+K)","Ca/(Na+Ca+K)","K/(Na+Ca+K)"),
end.member.names=c("Ab","An","Or") # Simplified
```

Note that the formulae can refer to valid atom names (see minFormula) and/or the names of the atoms allocated to the crystallographic sites (see minAllocateAtoms). In addition can be used any special parameters calculated by minValues, e.g.:

```
values.formulae=c("FeII/(Ca+Mg+FeII+Mn)", "Mg/(Ca+Mg+FeII+Mn)",
    "Mn/(Ca+Mg+FeII+Mn)"),
values.names=c("XFe", "XMg", "XMn"),
end.member.formulae=c("XMg", "XFe", "XMn", "FeIII/2"),
end.member.names=c("Prp","Alm","Sps","Adr") # Simplified
```

As an alternative, 'end.members.formulae' may contain a reference to an external file with R code that should reside in the main directory of *GCDkit.Mineral*. For example:

```
end.members.formulae="pyroxene.end.r",
end.member.names=c("Wo","En","Fs","Ac")
```

The file can contain any R language elements, the most useful being arguably the if/else conditions. The only restriction is that the script should return a matrix 'end.members' with a number of columns corresponding to the length of 'end.members.names' and a number of rows to the number of samples.

'minEndMembers' and 'minEndMembersAll' perform the calculations for all the analyses of the single mineral or the whole dataset, respectively.

Value

'minEndMembers' returns a numeric matrix with the calculated proportions of end members for the given mineral.

'minEndMembersAll' assigns the end-members, calculated for each of the minerals by the function 'minEndMembers', to the slot 'end.members' of the list 'min.data'.

Remark

The function minEndMembersC contains code identical to minEndMembers, compiled for speed.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class

minAssign

minCheckValency

minFormula

minAllocateAtoms

minValues

minClassify

minMain

Examples

```
data(combined)
accessVar(combined)
out<-minEndMembers(min.data$feldspar)
print(out)

out<-minEndMembersAll()
print(out)
```

---

mineral-class *Data storage and recalculation options*

Description

This is a default definition of slots available for any mineral species, further specified in the database (see details).

Details

The recalculation options for individual mineral species (classes) are stored in a default database, i.e. the file 'mineral_db.r', which is read always at the startup of the GCDkit.Mineral. The slots not specified explicitly in the database are taken as empty.

Slots

RECALCULATION OPTIONS

For *minAssign*:

full: character: all possible full names of the given mineral

abbreviated: character: abbreviations for the given mineral

For *minFormula*:

oxygens: numeric: (optional) number of oxygen equivalents the formula should be recalculated to.

atoms.sum: numeric: (optional) number of atoms the formula should be recalculated to.

atoms.recalc.list: character: (optional) list of atoms, if recalculation to a fixed sum of certain atoms is desired

charges: numeric: (optional) number of charges the formula should be recalculated to.

`cations`: numeric: (optional) number of cations in the formula, when Fe^{II}/Fe^{III} is to be estimated by the methods `Droop` (according to *Droop 1987*; referring to the total sum of cations) or by `FixedCats` (sum of the site specified by `'cations.site'`)

`cations.site`: character: (optional) name of the site, which should be summed by iterative iron estimation (`FixedCats`)

`iron`: character: (optional) Fe^{II}/Fe^{III} estimation method,
implemented are `'Droop'`, `'FixedCats'`, `'allFeII'` and `'allFeIII'`.
For amphiboles can be used also `'8Si'`, `'16CAT'`, `'15eNK'`,
`'15eK'`, `'13eCNK'`, `'8SiAl'`, `'10sumFeIII'` and `'avg'`; for pyroxenes also `'PxPapike'`

`atom.names`: character: names and order of atom names to be returned by formula recalculation (ideally, i.e. not all of them need to be present in the data file)

For *minAllocateAtoms*:

`sites`: list: each named component contains atoms that should be allocated to the given crystallographic site

`site.sums`: numeric: sums of individual sites, or NA when not needed/known

For *minValues*:

`values.formulae`: character: formulae for calculation of additional parameters (or a name of an external R script)

`values.names`: character: their names

For *minEndMembers*:

`end.member.formulae`: character: formulae for calculation of end-members (or a name of an external R script)

`end.member.names`: character: their names

DATA

`rough`: data.frame: the original analyses wt.% as imported from the data file (checked and possibly amended by `minCheckValency`)

`labels`: data.frame: all at least partly textual information on individual analyses from the file‘ plotting attributes (plotting symbols, their colours, size)

`recalc`: numeric matrix: analyses recalculated to atoms per formula unit (apfu)

`formula`: numeric matrix: apfu allocated to individual sites

`values`: numeric matrix: computed extra parameters

`end.members`: numeric matrix: calculated molar proportions of end members

`FeAmph`: numeric matrix: factors for Fe^{II}/Fe^{III} estimation by different methods - amphiboles only

Methods

formulaFixedOxygens `signature(object = "mineral")`: ...

formulaFixedAtoms `signature(object = "mineral")`: ...

formulaFixedCharges `signature(object = "mineral")`: ...

Droop `signature(object = "mineral")`: ...

allFeII `signature(object = "mineral")`: ...

allFeIII `signature(object = "mineral")`: ...

FixedCats `signature(object = "mineral")`: ...

PxPapike `signature(object = "mineral")`: ...

FeAmph `signature(object = "mineral")`: ...

minAllocateAtoms `signature(object = "mineral")`: ...

minCheckValency `signature(object = "mineral")`: ...

minFormula `signature(object = "mineral")`: ...

minValues `signature(object = "mineral")`: ...

minEndMembers `signature(object = "mineral")`: ...

print `signature(x = "mineral")`: ...

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Droop GTR (1987) A general equation for estimating Fe^{3+} concentrations in ferromagnesian silicates using stoichiometric criteria. Min Mag 51: 431-435 doi: 10.1180/minmag.1987.051.361.10doi: 10.1180/minmag.1987.051.361.10

Papike JJ, Cameron KL, Baldwin K (1974) Amphiboles and pyroxenes: characterization of other than quadrilateral components and estimates of ferric iron from microprobe data. Geol Soc Amer Abstr Prog 6: 1053-1054

See Also

`minAssign`

`minCheckValency`

`minFormula`

`minAllocateAtoms`

`minValues`

`minEndMembers`

`minClassify`

`minMain`

Examples

```
x<-names(getClass("mineral")@subclasses)
print(x)                                        # list of mineral classes

sapply(x,.getClassPrototype,slot="full")        # all full names

sapply(x,.getClassPrototype,slot="abbreviated") # all abbreviations

x<-new("garnet")

slotNames(x)
x@full
x@abbreviated
```

```
x@oxygens
x@iron
x@cations
```

`mineral.db-class` *Recalculation options for a single mineral*

Description

Object of this class, returned by the function `.formatDb`, contains recalculation options for a single mineral.

Details

The recalculation options for individual mineral species (classes) are stored in the default database, i.e. the file `'mineral_db.r'`, which is read always at the startup of *GCDkit.Mineral*. The slots not specified explicitly in the database are taken as empty.

Both methods, '`print`' and '`format`', serve to print all recalculation options available in the database. If no mineral is specified, the user has a choice from a drop-down list of the minerals present in the current dataset.

Slots

`dbase`**:** Object of class `"list"` with all the information ready for printing.

Methods

print `signature(x = "mineral.db")`: ...

format `signature(x,warning = TRUE)`: ...

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

`mineral-class`

Examples

```
options("gcd.shut.up"=TRUE)
data(combined)
# Automatic default recalculation, as specified in the database
accessVar(combined)

print(min.data$garnet)
options("gcd.shut.up"=FALSE)
```

`minFormula` *Recast mineral analyses to atoms per formula unit (apfu)*

Description

These functions serve for recalculation of electron microprobe analyses to atoms per formula unit (apfu) on the basis of fixed number of atoms, charges or oxygen equivalents (with or without F and Cl) equivalents.

Usage

```
formulaFixedOxygens(object)
formulaFixedOxygensC(object)
formulaFixedAtoms(object)
formulaFixedAtomsC(object)
formulaFixedCharges(object)
formulaFixedChargesC(object)
minFormula(object)
minFormulaAll()
Droop(object, recalc)
FixedCats(object, recalc)
allFeII(object, recalc)
allFeIII(object, recalc)
PxPapike(object, recalc)
FeAmph(object,recalc,fe.method)
```

Arguments

`object`	of the class "mineral"; data and recalculation options for a single mineral
`recalc`	numeric matrix with the recalculated data (apfu) for a single mineral
`fe.method`	one of the Fe^{II}/Fe^{III} recalculation options for amphibole, see below

Details

The function `'formulaFixedOxygens'` recalculates the chemical analyses to a given number of oxygen equivalents as specified, for each mineral class, in the slot `'oxygens'`. If oxygen is not the only anion present, the functions can handle halogens `'F'` and/or `'Cl'` as well (*Deer et al. 1996)*.

The function `'formulaFixedCharges'` recalculates the chemical analyses using a charge balance method. The number of desired charges is specified, for each mineral class, in the slot `'charges'`. For analyses in which either FeO or Fe_2O_3 has not been determined, Fe^{II}/Fe^{III} estimation is carried out in attempt to balance the formula precisely. For analyses in which this cannot be achieved, a warning message about their non-stoichiometry is displayed.

If no value is specified in the slots `'oxygens'` and `'charges'`, recalculation to a fixed number of atoms (taken from the slot `'atoms.sum'`) is performed. Then the sum of atoms can refer to the whole formula, or selected elements (if `atoms.recalc.list` is set).

`'minFormula'` and `'minFormulaAll'` are front-end functions that perform the mineral recalculation for all the analyses of the single mineral or the whole dataset, respectively.

The Fe^{II}/Fe^{III} ratio can be assessed (these methods are available always for all minerals) by one of the following methods:

Parameter	Method assumes
Droop	fixed no. of cations in the whole formula (*Droop 1987*)
FixedCats	fixed no. of cations in a single site (calculated iteratively)
allFeII	divalent iron only
allFeIII	trivalent iron only

The method `'Droop'` of *Droop (1987)* adjusts the Fe^{II}/Fe^{III} ratio assuming a fixed number of cations in the whole formula (as specified in the slot `'cations'` of the database).

The method `'FixedCats'` on the other hand estimates the Fe^{II}/Fe^{III} ratio setting the sum of a single crystallographic position (its name being given in `'cations.site'`) to the value specified by `'cations'`. This needs to be done iteratively, so the calculation tends to be rather time consuming.

Approach of *Papike et al. (1974)* for pyroxene recalculations is implemented in the method `'PxPapike'`. This procedure has been favoured, for instance, in thermobarometric formulae of *Putirka (2008)*.

For amphiboles are available also (through the function `'FeAmph'`) (*Leake et al. 1997, 2003*):

Parameter	Method assumes
8Si	8 Si atoms
16CAT	16 cations in the whole formula (equivalent to `'Droop'`)
15eNK	15 cations without Na and K
15eK	15 cations without K
13eCNK	13 cations without Ca, Na and K
8SiAl	sum Si + Al = 8
10sumFeIII	sum of Si + Al + Ti + Cr + Fe^{III} = 10
avg	average of the minimal and maximal estimates

Value

`'formulaFixedOxygens'` and `'formulaFixedAtoms'` return the recalculated formulae (atoms per formula unit, apfu) for the given mineral class. No Fe^{II}/Fe^{III} estimation is carried out.

`'formulaFixedCharges'` returns the recalculated formulae (atoms per formula unit, apfu) for the given mineral class. Fe^{II}/Fe^{III} estimation is carried out for analyses in which either FeO or Fe_2O_3 has not been determined.

`'minFormula'` returns a numeric matrix with the recalculated numeric data (apfu) for the given mineral class. The columns in the matrix are named by (subset of) atom names specified in the slot `'atoms'`. If required, the di- and trivalent iron are calculated, being labeled `'FeII'` and `'FeIII'`, respectively.

`'minFormulaAll'` assigns all the data (apfu) recalculated by `'minFormula'`, for each mineral separately, to the slots `'recalc'` of the list `'min.data'`.

`'Droop'` and `'FixedCats'` yield a single numeric vector with estimated Fe^{III} contents (apfu).

`'allFeII'`,`'allFeIII'` and `'FeAmph'` return a list with components:

`fe3`	Fe^{III} (apfu) estimated according to the preferred method.
`fe2`	Fe^{II} (apfu) estimated according to the preferred method.
`rf`	Fe^{II}/Fe^{III} recalculation factors (for internal use).

Remark

The functions `formulaFixedOxygensC`, `formulaFixedAtomsC`, and `formulaFixedChargesC` contain code identical to `formulaFixedOxygens`, `formulaFixedAtoms`, and `formulaFixedCharges`, compiled for speed.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (1996) An Introduction to the Rock-Forming Minerals. Prentice Hall, Harlow, 720 pp

Droop GTR (1987) A general equation for estimating Fe^{3+} concentrations in ferromagnesian silicates using stoichiometric criteria. Min Mag 51: 431-435 doi: 10.1180/minmag.1987.051.361.10doi: 10.1180/minmag.1987.051.361.10

Leake BE, Woolley AR, Arps CES, Birch WD, Gilbert MC, Grice JD, Hawthorne FC, Kato A, Kisch HJ, Krivovichev VG, Linthout K, Laird J, Mandarino J, Maresch WV, Nickel EH, Rock NMS, Schumacher JC, Smith JC, Stephenson NCN, Whittaker EJW, Youzhi G (1997) Nomenclature of amphiboles: report of the Subcommittee on Amphiboles of the International Mineralogical Association Commission on New Minerals and Mineral Names. Min Mag 61: 295-321 doi: 10.1180/minmag.1997.061.405.13doi: 10.1180/minmag.1997.061.405.13

Leake BE, Woolley AR, Birch WD, Burke EAJ, Ferraris G, Grice JD, Hawthorne FC, Kisch HJ, Krivovichev VG, Schumacher JC, Stephenson NCN, Whittaker EJW (2003) Nomenclature of amphiboles: additions and revisions to the International Mineralogical Association's 1997 recommendations. Can Min 41: 1355-1362 doi: 10.1180/0026461046810182doi: 10.1180/0026461046810182

Papike JJ, Cameron KL, Baldwin K (1974) Amphiboles and pyroxenes: characterization of other than quadrilateral components and estimates of ferric iron from microprobe data. Geol Soc Amer Abstr Prog 6: 1053-1054

Putirka KD (2008) Thermometers and barometers for volcanic systems. In: Putirka KD, Tepley III FJ (eds) Minerals, Inclusions And Volcanic Processes. Rev Mineral Geochem 69: 61-120 doi: 10.2138/rmg.2008.69.3doi: 10.2138/rmg.2008.69.3

See Also

mineral-class

minAssign

minCheckValency

minAllocateAtoms

minValues

minEndMembers

minClassify

minMain

Examples

```
x<-names(getClass("mineral")@subclasses)

sapply(x,.getClassPrototype,slot="oxygens")  # all oxygen equivalents

sapply(x,.getClassPrototype,slot="charges")  # all charges

sapply(x,.getClassPrototype,slot="atoms")    # all atoms in formulae

sapply(x,.getClassPrototype,slot="iron")     # all iron recalc options
```

```
sapply(x,.getClassPrototype,slot="cations")  # all sums of cations

# Garnet only
x<-new("garnet")
x@oxygens
x@atoms
x@iron
x@cations
```

`minMain` *Mineral analyses recalculation*

Description

This mother-of-all-recalculations recasts the mineral analyses obtained by electron microprobe to atoms per formula unit (apfu). If desired it also allocates the atoms to individual crystallographic sites, recasts the analyses to end members and/or calculates some extra parameters.

Usage

```
minMain(mineral="",min.slots=list())
```

Arguments

`mineral`	character; the name of the mineral class.
`min.slots`	list of recalculation options.

Details

The complete recalculation procedure is as follows (in reality, some of the steps may be skipped):

1. atoms per formula unit: `minFormula`
2. allocation of atoms to crystallographic sites: `minAllocateAtoms`
3. extra parameters: `minValues`
4. end members: `minEndMembers`

Value

All the original and recalculated data are stored in a list `'min.data'` with named components for each of the mineral species (e.g. `'min.data$garnet'`). See `mineral-class` for details. Additionally the function returns a list `'results'` with complete recalculated mineral analyses.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class

minAssign

minCheckValency

minFormula

minAllocateAtoms

minValues

minEndMembers

minClassify

Examples

```
x<-names(getClass("mineral")@subclasses)
print(x)

# 1 Direct assignment to a variable #
#####################################
## Not run:
min.data$garnet<-new("garnet")
x<-matrix(c(38.83,0.34,19.43,6.50,NA,1.15,0.22,32.91,
    38.89,0.18,20.33,4.92,NA,0.66,0.24,33.88,
    38.24,0.38,17.97,8.39,NA,1.02,0.18,32.73),byrow=T,nrow=3)
colnames(x)<-c("SiO2","TiO2","Al2O3","FeO","Fe2O3","MnO","MgO","CaO")
accessVar(x)

minFormulaAll()

min.data$garnet@rough        # original data

min.data$garnet@recalc       # recalculated apfu

min.data$garnet@formula      # assigned crystallographic sites

min.data$garnet@end.members  # end members

## End(Not run)

# 2 On demand recalculation from a file/sample dataset #
########################################################
# Automatic default recalculation, as specified in the database
data(combined)
accessVar(combined)

# Garnet 12 O
minMain("garnet",list(oxygens=12))

# Garnet 12 O, FeII/FeIII allocation according to Droop (1987)
minMain("garnet",list(oxygens=12,cations=8,iron="Droop"))

# Garnet 12 O, assuming 2 cations in Y site
minMain("garnet",list(oxygens=12,cations=2,cations.site="Y",iron="FixedCats"))

# Garnet 8 cations
```

```
minMain("garnet",list(atoms.sum=8))

# Clinopyroxene 6 O, FeII/FeIII allocation according to Papike et al. (1974)
minMain("clinopyroxene",list(oxygens=6,iron="PxPapike"))
```

`minValues` *Calculation of extra parameters*

Description

Calculates extra values specified by formulae stored in the database or external R scripts.

Usage

```
minValues(object)
minValuesAll()
```

Arguments

`object`	of the class "mineral"; data and recalculation options for a single mineral

Details

The additional parameters can be defined in the database (in the slots `'values.formulae'` and `'values.names'`), similar to the following example:

```
values.formulae=c("Ca/(Ca+Mg+FeII+Mn)", "FeII/Mg", "Al_Z/Al_Y"),
values.names=c("XCa","Fe/Mg","AlIV/AlVI")
```

This means that the formulae can refer to valid atom names (as are the first two above) and/or the names of the atoms allocated to the individual crystallographic sites by `minAllocateAtoms` (the third one).

`'minValues'` and `'minValuesAll'` perform the calculations for all the analyses of the single mineral or the whole dataset, respectively.

As an alternative, `'values.formulae'` may contain a reference to an external file with R code that should reside in the main directory of *GCDkit.Mineral*. For example:

```
values.formulae="pyroxene.r",
values.names=c("FeIII/Fetot","XMg","AlIV/AlVI","AlIV","AlVI"),
```

The file can contain any R language elements, the most useful being arguably the if/else conditions. The only restriction is that the script should return a matrix `'values'` with a number of columns corresponding to the length of `'values.names'` and a number of rows to the number of samples.

Value

`'minValues'` returns a numeric matrix with the calculated parameters for the given mineral class.

`'minValuesAll'` assigns the calculated parameters, for each mineral separately, to the slot `'values'` of the list `'min.data'`.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

mineral-class

minAssign

minCheckValency

minFormula

minAllocateAtoms

minEndMembers

minClassify

minMain

Examples

```
data(combined)
accessVar(combined)
out<-minValues(min.data$garnet)
print(out)

out<-minEndMembersAll()
print(out)
```

Molecular weights *Calculating molecular weights of oxides*

Description

Calculates molecular weights for simple oxide formula(e).

Usage

```
molecularWeight(formula)
```

Arguments

formula	a character vector with formula(e) of the oxide(s).

Details

So far only simple oxide formulae in form of A_xO_y (where x, y are optional indexes) can be handled. The atomic weights are stored in a file MW.data in the main directory. The atomic weights come from official CIAAW web site https://www.ciaaw.org.

Value

A list with items:

MW	molecular weight(s)
x.atoms	number(s) of atoms in the formula
x.oxygen	number(s) of oxygens

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz> and Vojtěch Erban, <erban@sopky.cz>

References

Commission on Isotopic Abundances and Atomic Weights (CIAAW) of the International Union of Pure and Applied Chemistry. Accessed on January 8, 2016, at https://www.ciaaw.org

Examples

```
molecularWeight("SiO2")
molecularWeight("SiO2")[[1]]

MW["SiO2"]
MW[major]
mw["Si"]

oxides<-c("SiO2","TiO2","Al2O3","Fe2O3","FeO")
sapply(oxides,molecularWeight)
```

`Multiple plots` *Multiple binary plots*

Description

These functions plot multiple binary plots with a common x axis, such as Harker plots.

Usage

```
multiple(x, y = paste(colnames(WR), sep = ","),
    samples = rownames(WR), pch = labels$Symbol,
    col = labels$Colour, xmin = NULL, xmax = NULL,
    GUI = FALSE, nrow = NULL, ncol = NULL, title = NULL,...)

multipleMjr(x = "",
    y = "SiO2,TiO2,Al2O3,FeOt,MgO,CaO,Na2O,K2O,P2O5",
    pch = labels$Symbol, col = labels$Colour, ...)
multipleTrc(x = "",
    y = "Rb,Sr,Ba,Cr,Ni,La,Ce,Y,Zr,mg#,A/CNK,K2O/Na2O",
    pch = labels$Symbol, col = labels$Colour, ...)
```

Arguments

`x`	a character vector, name of the common x axis. Formulae are OK.
`y`	a character vector, names of oxides/elements to be plotted as y axes separated by commas. Formulae are OK.
`nrow, ncol`	dimensions of the plots' matrix

`samples`	character or numeric vector; specification of the samples to be plotted.
`pch`	plotting symbols.
`col`	plotting colours.
`xmin, xmax`	minimum and maximum for the x axis.
`GUI`	logical; is the call being made from within GCDkit GUI or not?
`title`	character; title for the plate of multiple plots
`...`	further graphical parameters: see 'help(par) for details.

Details

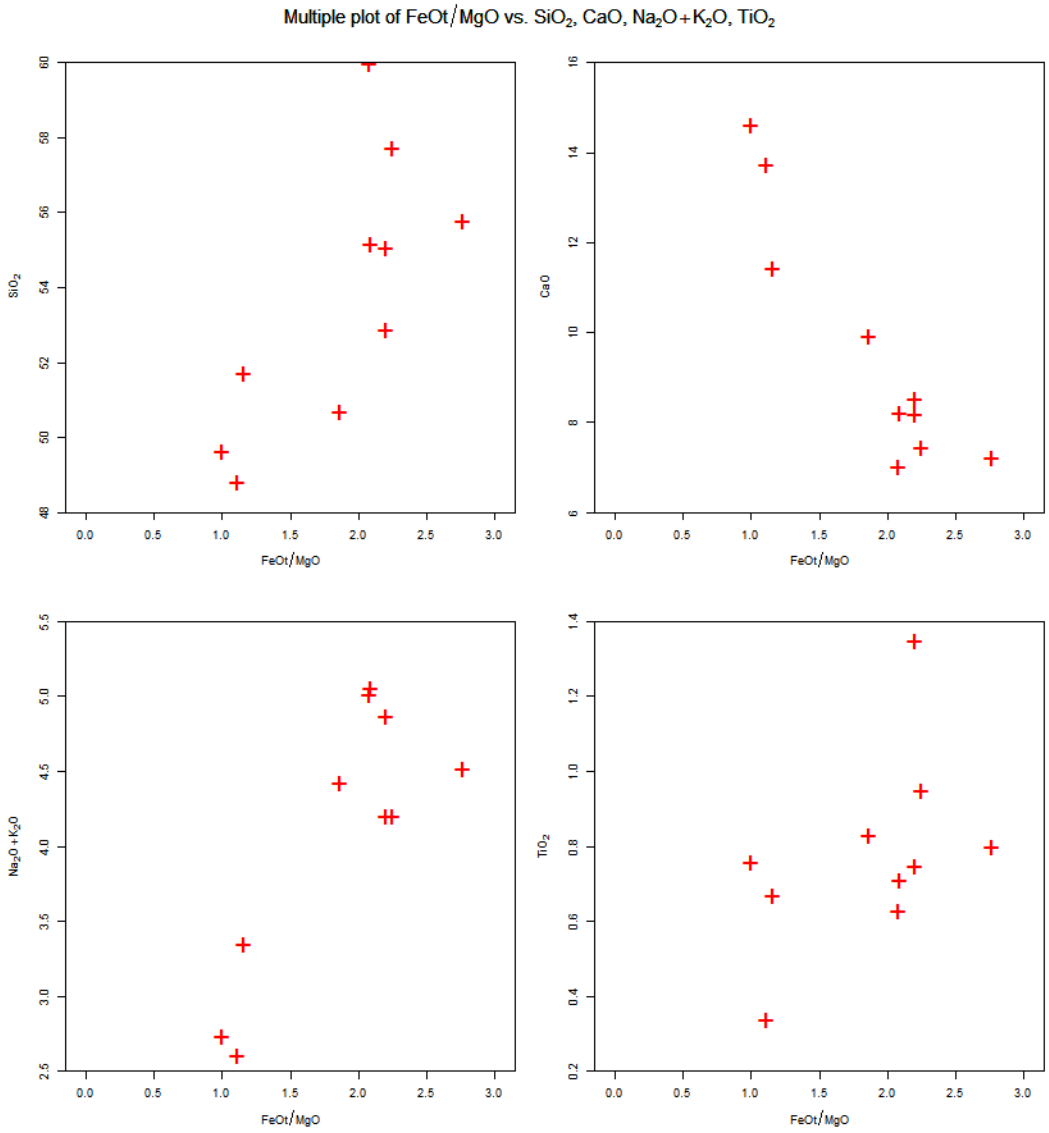
If x axis occurs among the arguments to be plotted as y axes, it is skipped.

Functions 'multipleMjr' and 'multipleTrc' are entry points supplying the default lists for major- and trace elements.

Even though as a default is assumed a list of major (SiO2,TiO2,Al2O3,FeOt,MnO,MgO,CaO, Na2O,K2O) or trace (Rb,Sr,Ba,Cr,Ni,La,Ce,Y,Zr and mg#) elements, the variable(s) to be displayed can be specified.

The easiest way is to type directly the names of the columns, separated by commas. Alternatively can be used their sequence numbers or ranges. Also built-in lists can be employed, such as 'LILE', 'REE', 'major' and 'HFSE' or their combinations with the column names.

These lists are simple character vectors, and additional ones can be built by the user (see Examples). Note that currently only a single, stand-alone, user-defined list can be employed as a search criterion.



In the specification of the x axis or any of the y axes can be used also arithmetic expressions, see `calcCore` for the correct syntax.

Lastly, the user is asked to enter the limits for the x axis, two numbers separated by a comma. Note that the scaling takes into account the size of the plotting symbols, i.e. the axes are extended somewhat.

Value

None.

Note

This function uses the plates concept. The individual plots can be selected and their properties/appearance changed as if they were stand alone Figaro-compatible plots. See `Plate`, `Plate editing` and `figaro` for details.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

figaro, Plate, Plate editing

Examples

```
data(sazava)
accessVar("sazava")

# Harker plots
multipleMjr("SiO2")

# Plots the LILE against the sum of alkalis
multiple("Na2O+K2O",LILE,xmin=0)

multiple("FeOt/MgO","SiO2,CaO,Na2O+K2O,TiO2",pch="+",col="red",samples=1:10,cex=2.5)

# Plots the default trace-element set against the Zr, prettified
multipleTrc("Zr")
plateCex(1.8)
plateCexLab(1.5)
plateXLim(c(0,200))
```

---

olivine *Selected olivine analyses*

Description

This data set gives the selected chemical analyses of olivine from *Deer et al. (2013)*.

Usage

```
data(olivine)
```

Format

A data frame containing 6 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 8 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(olivine)
accessVar("olivine")
print(WR)
```

`oxide2oxide` *Recalculation of one oxide to a different one*

Description

Returns a factor needed to multiply concentrations of an element given as an oxide (in wt %) to a different target oxide (of the same element).

Usage

```
oxide2oxide(formula1, formula2)
```

Arguments

`formula1`	character: the oxide which is to be recalculated
`formula2`	character: the target oxide

Value

A factor for recalculation.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

`oxide2ppm`, `ppm2oxide`, `molecularWeight`

Examples

```
oxide2oxide("FeO","Fe2O3")
oxide2oxide("Mn2O3","MnO")
```

oxide2ppm *Calculation of ppm of atom from wt% of an oxide*

Description

Recasts concentrations of an oxide (in wt. %) to that of appropriate cation (in ppm).

Usage

```
oxide2ppm(formula,where="WR")
```

Arguments

formula	character: the oxide which is to be recalculated
where	character: a name of matrix or dataframe with the data to be recalculated

Value

A numeric matrix with one column containing the recalculated concentrations of the given cation (ppm) for individual samples.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

ppm2oxide, oxide2oxide, molecularWeight

Examples

```
data(sazava)
accessVar("sazava")

oxide2ppm("K2O")

oxide2ppm("FeOt")
oxide2ppm("FeO")+oxide2ppm("Fe2O3")
```

pairsCorr *Statistics: Correlation*

Description

Plots a matrix of scatterplots in the lower panel and one of other pre-defined panel functions in the upper.

Usage

```
pairsCorr(elems=NULL,pch=NULL,col=NULL,upper.panel="panel.cor",...)

pairsMjr(pch=NULL,col=NULL,upper.panel="panel.cor",...)

pairsTrc(pch=NULL,col=NULL,upper.panel="panel.cor",...)
```

Arguments

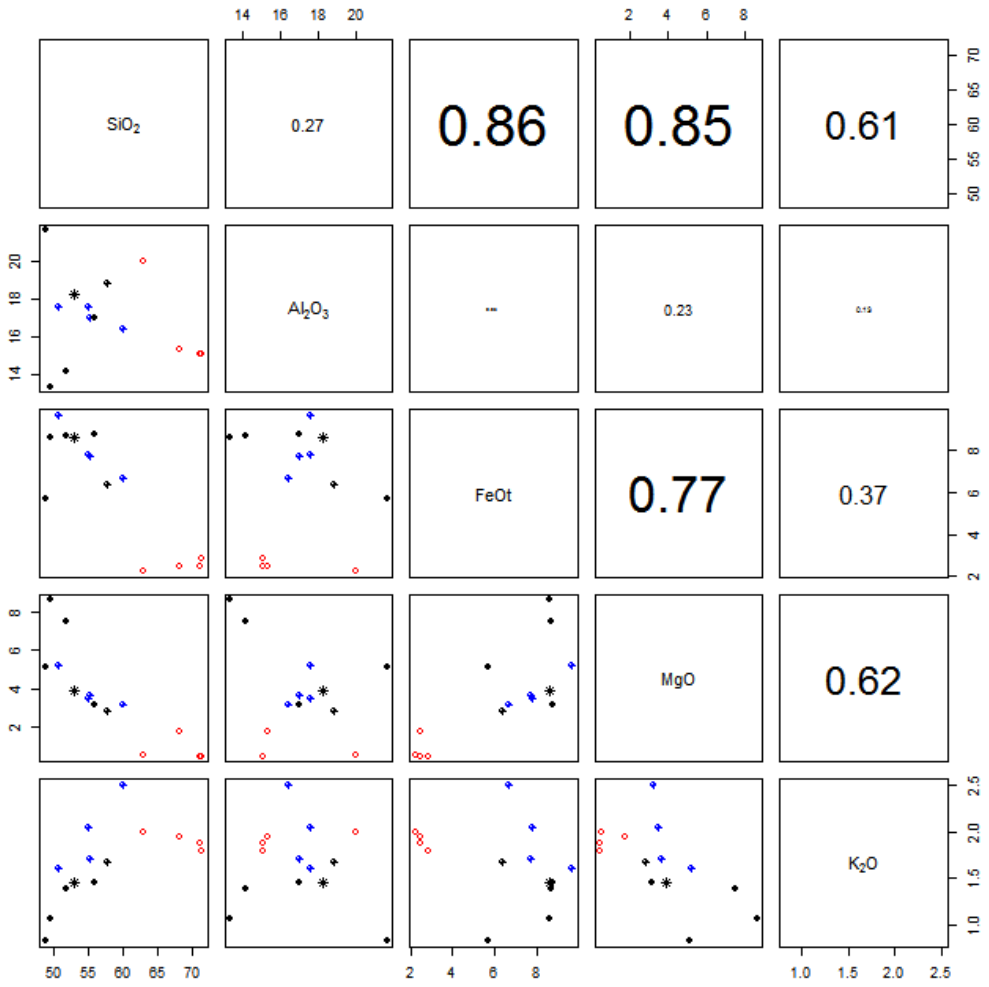
`elems`	list of desired elements/oxides
`pch`	plotting symbols
`col`	colours
`upper.panel`	character; name of the function to be used for the upper plate
`...`	any further parameters to the function `pairs`

Details

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSamples` for details.

Even though a list of major elements is assumed as a default, different variables can be specified by the function '`selectColumnsLabels`'.

In the specification of the variables can be used also arithmetic expressions, see `calcCore` for the correct syntax.



The upper panels to choose from are:

Parameter	Function	Explanation
'panel.corr'	cor	Prints correlations, with size proportional to the correlations;
'panel.cov'	cov	Prints covariances;
'panel.smooth'	lowess	Fits smooth trendlines;
'panel.hist'	hist	Plots frequency histograms.

Value

Returns a variable 'results' with the outcome of the function used for the upper panel construction (cor for 'panel.corr' or cov for 'panel.cov').

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

pairs cor cov lowess hist

Examples

```
data(blatna)
accessVar("blatna")

pairsCorr(LILE)

pairsCorr(LILE,pch="+",col="darkgreen",cex.labels=3,upper.panel="panel.smooth")

pairsMjr(pch=15,col="red")

pairsTrc(pch=15,col="red")

# user-defined list
my.elems<-c("Na2O/K2O","Rb","Sr","Ba")
pairsCorr(my.elems,col="darkblue")
```

---

pdfAll *Save all graphics to PDF*

Description

Saves all graphical windows to a single PDF file.

Usage

```
pdfAll(filename=NULL)
```

Arguments

filename	a name of file for saving the output.

Details

The function prompts for filename under which it saves all graphical windows, each on a separate page. PDF is the most portable format, that should preserve practically the same layout on all platforms.

Individual diagram can be saved from a menu that appears after clicking on the appropriate graphical window ('File|Save as|PDF').

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'psAll' 'pdf'

peekDataset *Retrieving previous dataset stored in memory*

Description

Both functions restore the previously stored dataset and make it current.

Usage

```
peekDataset(which.dataset=NULL)
selectDataset()
```

Arguments

which.dataset	numeric or character; a sequence number or name of the stored dataset.

Details

The function 'peekDataset' restores a dataset saved previously into memory by the function 'pokeDataset'. This means that it assigns all global variables specified by individual items of the list 'WRCube'.

These typically are: 'WR', 'WRanh', 'milli', 'labels', 'filename', 'groups' and 'grouping'.

The function 'selectDataset' provides a graphical interface to 'peekDataset', i.e. shows a list box filled by the names of datasets currently stored in the memory.

Value

None. But several global variables, among others 'WR', 'WRanh', 'milli' and 'labels', are affected. The name of the current dataset is stored in 'dataset.name'.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'pokeDataset' 'purgeDatasets'

Examples

```
data(sazava)
accessVar("sazava")
# stored as sazava in WRCube

assignColVar("MgO","blues")
assign1symb(15)
# store a new copy in the WRCube
pokeDataset("coloured sazava")

data(swiss)
accessVar("swiss")
# stored as swiss in WRCube
```

```
peekDataset("sazava")
binary("SiO2","Ba")

peekDataset("coloured sazava")
binary("SiO2","Ba")

peekDataset("swiss")
binary("Catholic","Education",pch=15,col="darkgreen")

# Second dataset within the WRCube
names(WRCube)

peekDataset(2)
binary("SiO2","Sr")
```

`peterplot` *Anomaly plot*

Description

This function plots a conventional binary diagram but the type and size of the plotting symbols is assigned according to the distribution of a third, conditioning variable.

Usage

```
peterplot(xaxis = "", yaxis = "", zaxis = "", ident = FALSE,
    scaling.small = labels[1,"Size"], scaling.big = 2 * scaling.small,
    assign.symbols = FALSE)
```

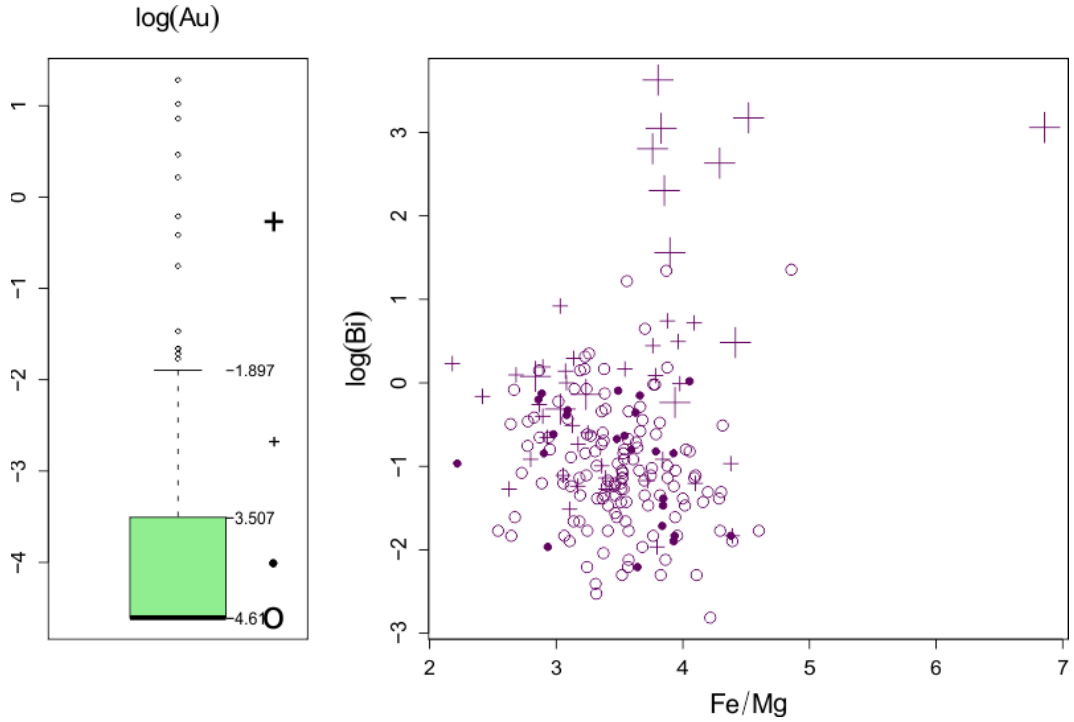
Arguments

`xaxis, yaxis`	character; specification of the axes
`zaxis`	character; conditioning variable
`ident`	logical; identify the individual points?
`scaling.small`	scaling factor for the smaller plotting symbols
`scaling.big`	scaling factor for the larger plotting symbols
`assign.symbols`	logical; should be the plotting symbols and their sizes assigned permanently?

Details

If no parameters `xaxis,yaxis` and `zaxis` are specified, the user is prompted to do so interactively.

The plotting symbols are assigned as follows: the values within 25 quartiles) obtain a `dot`, the higher ones are denoted by '+'and lower ones by '-'. If the given value is an outlier, its plotting size is doubled.



Optionally, the user can assign the plotting symbols and their sizes permanently, for use in other diagrams throughout the system.

Value

May modify the variable `cex`, as well as the codes of plotting symbols stored in the data frame `labels`.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Reimann C, Filzmoser P, Garrett RG (2002) Factor analysis applied to regional geochemical data: problems and possibilities. Applied Geochemistry 17: 185-206

Examples

```
data(sazava)
accessVar("sazava")

peterplot("SiO2","MgO","K2O")

# Permanent assignment of plotting symbols based on peterplot
peterplot("SiO2","MgO","K2O",assign.symbols=TRUE)
binary("SiO2","Na2O+K2O")
```

phasePropPlot *Stacked barplot of temperature vs. phase proportions.*

Description

This function makes a stacked barplot of phase proportions, typically of minerals with, or without, melt.

Usage

```
phasePropPlot(mat, renormalize = TRUE, col = NULL, palette = "jet.colors",
leg.pos = "bottomleft", leg.bg = "#FFFFFFAA", xlab = expression(Temperature~degree*C),
ylab = "Phase proportions", xlim = NULL, ylim = c(0, 1), border = "white", main="")
```

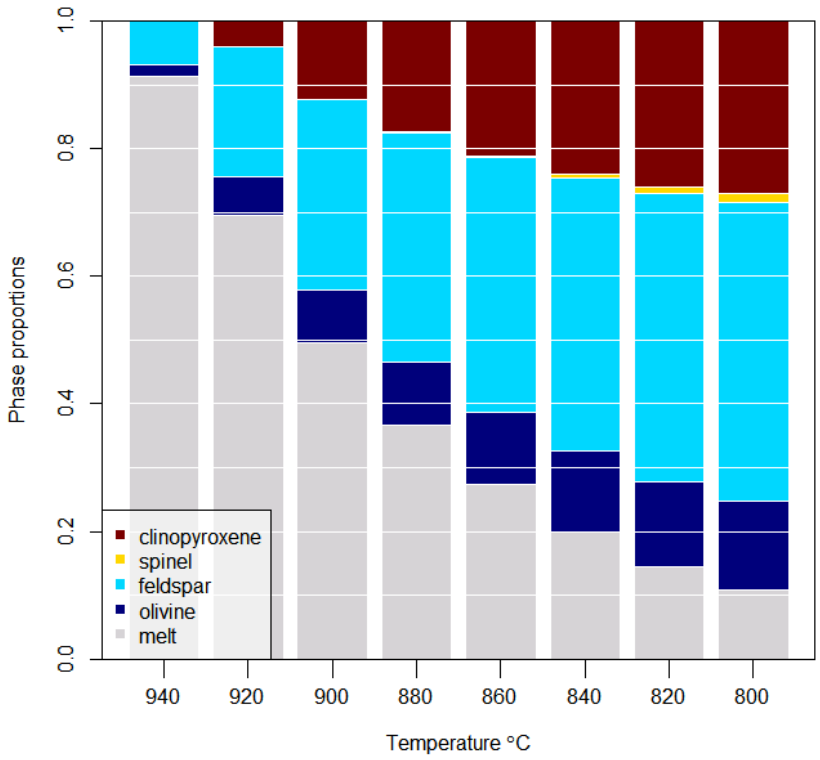
Arguments

mat	a numeric matrix with phase proportions in columns, and temperature in C in rows.
renormalize	logical, should be the data in mat renormalized to a sum of 1?
col	list of colours for each of the phases.
palette	palette name.
leg.pos	position of the legend.
leg.bg	background colour for the legend.
xlab	character or expression; label for the x axis
ylab	character or expression; label for the y axis.
xlim	limits for the x axis.
ylim	limits for the y axis.
border	colour for the border for each of the bars.
main	character; main title for the plot.

Details

The input is a matrix with phase proportions in columns, their names in colnames and variable (by default a temperature in Â°C) in rownames.

If 'col = NULL' and 'palette' is specified, then the corresponding number of colours are taken therefrom. Then the first column of data, typically a melt, is shown in gray.



The function assigns data for the diagram into a Figaro template (list 'sheet'), centers of intervals into 'x.data' (not used for the x axis labeling) and the plotting matrix into 'y.data'. The values for labeling the x axis are taken from rownames of 'y.data'.

Value

sheet	list with Figaro Style Sheet data.
x.data	See Details.
y.data	See Details.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

figaro

Examples

```
min.prop<-matrix(c(0.1,0.2,0.5,
                   0.2,0.25,0.25,
                   0.5,0.4,0.15,
                   0.2,0.15,0.1),
    nrow=3,ncol=4,dimnames=list(seq(750,850,by=50),c("Liq","Cpx","Opx","Pl")))
```

```
phasePropPlot(min.prop,palette="jet.colors", ylab="vol. percent",
    main="Plot of mineral proportions")

phasePropPlot(min.prop,col=1:4)

phasePropPlot(min.prop,col=heat.colors(4))

data(blatna)
accessVar("blatna")
windows(10,5)
i<-names(sort(WR[,"SiO2"]))
phasePropPlot(WR[i,major],xlab="Sample")
```

`Plate` *Plotting plates of several diagrams*

Description

Functions to set up, save or load a so-called 'plate', i.e. a regular grid of slots to accommodate (any mixture of) binary or ternary plots, spiderplots or such alike. For instance, Harker plots are implemented using the plate concept.

Usage

```
multiplePerPage(which=NULL,nrow=NULL,ncol=NULL,title="Plate",
    dummy=FALSE,new=TRUE)

Plate(scr=NULL)

plateRedraw(device="windows",filename=NULL,colormodel="rgb")

platePS(colormodel="rgb",filename=NULL)

plateSave()

plateLoad()
```

Arguments

`which`	total number of slots to be occupied by individual diagrams.
`nrow`	number of rows in the plots' matrix.
`ncol`	number of columns in the plots' matrix.
`title`	title for the whole plate.
`dummy`	logical; if TRUE, dummy plots are shown. See Details.
`new`	(logical; should be a new plotting window opened?
`scr`	(optional) number of screen to be selected.
`device`	output device; either `'windows'` or `'postscript'`.
`filename`	name of file if output is to be redirected to Postscript.
`colormodel`	color mode for Postscript; `'rgb'` or `'gray'`.

Details

The function 'multiplePerPage' serves to setting up a matrix of slots, each of which could be taken by a single Figaro-compatible diagram (a binary plot, a ternary plot, a spiderplot,...). If 'which' is NULL, the function asks for their number, and then suggests number of rows ('nrow') and columns ('ncol') for the matrix arrangement.

If desired, the slots can be filled by the so-called 'dummy plots', i.e. gray boxes showing the exact position and the size of each of them.

If 'which' is an integer, specified number of slots is allocated. Alternatively, this argument may represent a vector containing any mixture of names of diagrams that can be plotted by the function plotDiagram or even plotting commands themselves used to fill the individual slots directly. See Examples.

Once set up, a single slot can be selected for further work using the function 'Plate'. The function can be called directly, with the number of the screen desired. If none is specified, a red box-like cursor appears in the graphical window, which can be moved around using the cursor keys, Spacebar or by mouse. The appropriate slot can be chosen by left mouse button or by pressing Enter. Right-click anywhere on the plate invokes a context menu which enables several actions:

Menu item	Function
Introduce plot	Select a new Figaro-compatible diagram for this slot.
Plot editing	Modify the existing diagram (like the menu Plot editing for stand alone plots).
Plate editing	Functions to modify the overall plate properties or all its diagrams simultaneously.

The function 'plateRedraw' serves for replotting a 'clean! version of the whole plate, eg. for saving/printing, For this purpose, its output can be redirected to Postscript, either in colour or as black and white. As a wrapper for the Postscript output serves the function 'platePS'

The functions 'plateSave' and 'plateLoad' are designed to save and retrieve definitions of plates (Figaro sheets and the relevant data) for later use. The default suffix for the saved plates is 'mgr'. Note that only the data needed for the plotting ('x.data', 'y.data') are stored in the 'mgr' files. Thus the data set currently in memory (e.g., variables 'WR', 'labels', ...) is unaffected by the function 'plateLoad'.

Starting with GCDkit version 3, the plates concept is used by some built-in functions, such as 'Multiple plots' (function multiple) or 'Multiple plots by groups' (function figMulti).

Value

plate	list of Figaro definitions for individual diagrams
plate.data	list containing 'x.data' and 'y.data' for each of them

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

Plate editing, multiple, figMulti, plot, binary, ternary, figaro

Examples

```
data(sazava)
accessVar("sazava")
```

```
multiplePerPage(
    c("binary('SiO2','Na2O',new=FALSE)",
    "binary('SiO2','K2O',new=FALSE)",
    "binary('SiO2','Na2O+K2O',new=FALSE)",
    "binary('SiO2','Rb',new=FALSE)")
)
plateUser(las=3,cex=2.5,col="darkred",cex.axis=1.5,cex.lab=1.8)

# Works only in GCDkit proper
Plate(3)
plotDiagram("LarochePlut",FALSE,FALSE)
plateRedraw()
```

`Plate editing` *Editing the plate properties/all its plots simultaneously*

Description

A collection of functions to modify the properties of a plate (or all its diagrams) simultaneously.

Usage

```
plateUser(...)

plateCex(n=NULL)

plateCexMain(n=NULL)

plateCexLab(n=NULL)

plateAnnotationsRemove()

platePch(pch=NULL)

plateCol(col=NULL)

plateBW()

plateXLim(xlim=NULL)

plateYLim(ylim=NULL)

plate0YLim()

plateExpand(scr=NULL)

plateExtract(diagram,which=NULL,main=NULL,calc.only=FALSE,...)
```

Arguments

`...`	(for `plateUser`): list of arguments passed to the function `'figUser'` (for `plateExtract`): additional parameters to the diagram (plate) plotting function.

`n`	relative size (use n = 1 for standard one).
`pch`	plotting symbol specification, either as string or a numeric code (showSymbols).
`col`	colour specification, either by its English name, or by a numeric code (showColours).
`xlim`	scaling for the x axis.
`ylim`	scaling for the y axis.
`scr`	number of screen to be expanded.
`diagram`	name of the plotting function producing a plate.
`which`	sequential number of plot in its definition.
`main`	optional alternative main title to the diagram.
`calc.only`	logical; should be performed only calculations, without plotting?

Details

Most of these functions serve to simultaneously change properties of all individual diagrams forming the given plate. They can be used to set up a uniform size of plotting symbols (`'plateCex'`), scaling the main title (`'plateCexMain'`), set up a uniform size of the axes' labels (`'plateCexLab'`), remove the annotations of classification fields (`'plateAnnotationsRemove'`), specifying a uniform plotting symbol (`'platePch'`) and/or colour (`'plateCol'`) to all plots, or set them into black and white (`'plateBW'`).

Several parameters can be changed at the same time, using the powerful function `'plateUser'`. It simply passes all its arguments to the function `'figUser'`, invoked for modification of each of the individual slots.

If the same variable is plotted as x or y axis in all diagrams forming the plate (e.g., on Harker plots), it can be scaled at once by means of the functions `'plateXLim'` and `'plateYLim'`. Using the command `'plate0YLim'` it is possible to set the origin of all non-logarithmic y axes to zero.

The function `'plateExpand'` displays an expanded version of the selected diagram in a separate window.

The function `'plateExtract'` extracts a Figaro definition of a single plot from a plate normally plotted by the function `'diagram'`. If `'calc.only'` is `'FALSE'`, the diagram is displayed, either in a separate window or in the current slot, if the active graphical window contains a plate.

Value

None.

Author(s)

Vojtěch Janoušek, `<vojtech.janousek@geology.cz>`

References

Pearce JA, Harris NW & Tindle AG (1984) Trace element discrimination diagrams for the tectonic interpretation of granitic rocks. J Petrology 25: 956-983. doi: 10.1093/petrology/25.4.956

See Also

Plate, figaro, figUser, par, figCol, figScale, showSymbols, showColours

Examples

```
data(sazava)
accessVar("sazava")

multiplePerPage(which=c("binary(\"K2O/Na2O\",
    \"Rb\",new=FALSE)","DebonPQ","DebonBMgNo","AFM","PeceTaylor","Shand"))
plateCex(0.5)

plateCex(2)
platePch(11)

platePch("+")
plateCol(11)

plateCol("red")

plateBW()

# For power users
plateUser(las=3,cex=2.5,col="darkred",cex.axis=1.5,cex.lab=1.8)

# Harkers
multiple("SiO2",major)
plateCex(2)
plateXLim(c(50,70))
plate0YLim()

# Spiders
groupsByLabel("Intrusion")
spider(WR,selectNorm("Boynton"),0.1,1000,pch=labels$Symbol,col=labels$Colour,cex=2)
figMulti(plot.symb=TRUE)
plateYLim(c(1,100))

plateExpand(2)

# Second plot of Pearce et al. (1984), i.e. Y-Nb
graphicsOff()
plateExtract("PearceGranite",2)
```

---

plateLabelSlots *Annotate individual slots by letters or Roman numerals*

Description

Annotates individual slots in a plate by letters or Roman numerals. For instance (a), (b), (c)... or (i), (ii), (iii), (iv), (v)...

Usage

```
plateLabelSlots(text=letters,style="()",cex=1.5,pos="topright")
```

Arguments

text	desired type of labels; see Details.
style	optional character strings before and after label, typically brackets.
cex	relative size of the text compared to the current codepar("cex").
pos	character; position of the label relative to the plot.

Details

The argument 'what' may acquire one of following values:

```
'letters' 'LETTERS' 'numbers' 'roman' 'ROMAN'
```

or can be user-defined character string of longer or of the same length as is the number of slots to be annotated (see the last example).

Possible positions (parameter pos) are:

```
'bottomright' 'bottom' 'bottomleft' 'left'
'topleft' 'top' 'topright' 'right' 'center'
```

.

Value

none

Note

This function uses the plates concept. The individual plots can be selected and their properties/appearance changed as if they were stand alone Figaro-compatible plots. See Plate, Plate editing and figaro for details.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

Plate, Plate editing, figaro

Examples

```
data(sazava)
accessVar("sazava")

multipleMjr("SiO2")
plateCex(1.8)
plateCexLab(1.5)

plateLabelSlots("letters","",pos="bottomleft")

plateLabelSlots("ROMAN","{}")

my_labs<-c("1st","2nd","3rd","4th","5th","6th","7th","8th","9th")
plateLabelSlots(my_labs)
```

plotWithCircles *xyz plotWithCircles*

Description

Plots a background binary diagram of two specified variables and the whole dataset or its selection. The size and colours of the plotted circles correspond to the third.

Usage

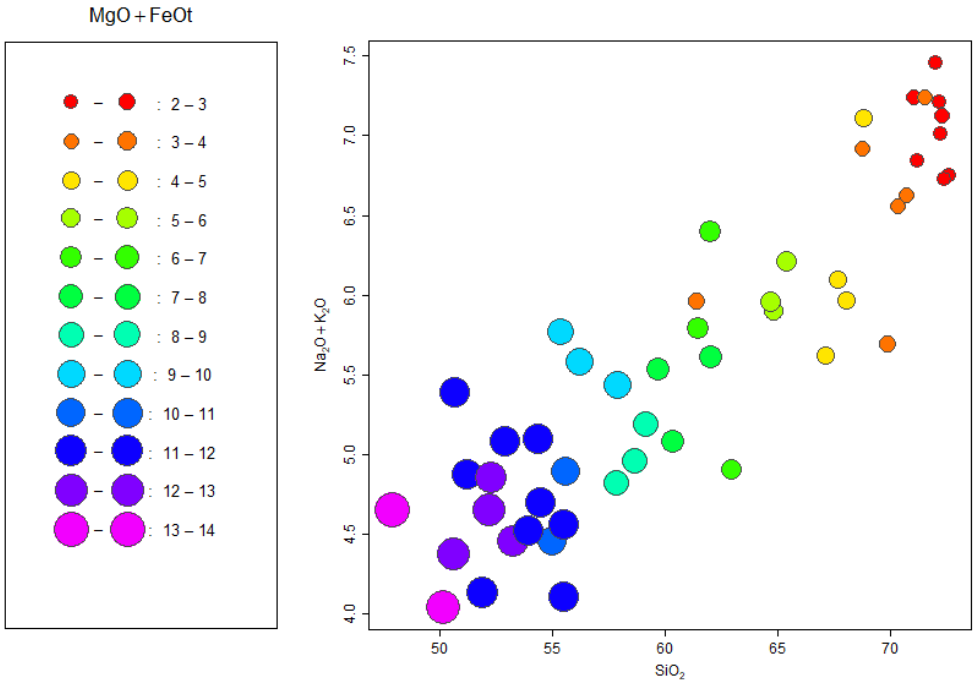
```
plotWithCircles(xaxis = "", yaxis = "", zaxis = "",
    colour = "heat.colors", scaling.factor = 1,
    bins = NULL, ident = getOption("gcd.ident"), alpha="FF")
```

Arguments

xaxis	Name of the data column to be used as x axis.
yaxis	Name of the data column to be used as y axis.
zaxis	Name of the data column to determine the size/colour of the circles.
colour	colour scheme for the circles.
scaling.factor	a factor determine the size of the circles.
bins	number of intervals for the legend.
ident	Logical: should be the individual samples identified?
alpha	hexadecimal number indicating the alpha channel (transparency).

Details

This function produces a custom binary plot, with the size and colours of the plotted circles corresponding to a third variable.



If no parameters `'xlab'`,`'ylab'` and `'zlab'` are given, the user is prompted to specify them.

The variables are selected using the function '`selectColumnLabel`.

In the specification of the apices can be used also arithmetic expressions, see `calcCore` for the correct syntax.

The samples to be plotted can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSubset` for details.

The legal colour schemes are: '`"grays"`,`"reds"`,`"blues"`,`"greens"`,`"cyans"`,`"violets"`,`"yellows"` '`"cm.colors"`,`"heat.colors"`,`"terrain.colors"`,`"topo.colors"`,`"rainbow"`,`"jet.colors"`'.

Optionally, the colours can be made semitransparent, if hexadecimal parameter `'alpha'` is specified for the alpha channel (transparency).

Value

None.

Warning

This function IS NOT Figaro-compatible.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

& Vojtěch Erban, <erban@sopky.cz>

Examples

```
data(blatna)
accessVar("blatna")
```

```
plotWithCircles("SiO2","Na2O+K2O","MgO+FeOt",colour="rainbow")

plotWithCircles("SiO2","MgO","K2O",colour="grays",scaling.factor=0.5,ident=TRUE)
```

---

pokeDataset *Storing a dataset into memory for later use*

Description

Saves the current dataset into memory so that it can be later re-stored.

Usage

```
pokeDataset(which.dataset=NULL,
   par.list="WR,WRanh,milli,labels,filename,groups,grouping,init,age,leg.pch,leg.col",
    overwrite.warn=TRUE)
```

Arguments

`which.dataset`	character; a name of the stored dataset.
`par.list`	list of global variables to be stored.
`overwrite.warn`	logical, warn if a dataset is going to be rewritten in 'WRCube'. See Details.

Details

This function stores the global variables specified by `par.list`, typically 'WR', 'WRanh', 'milli' 'labels', 'filename', 'groups' and 'grouping' into the list 'WRCube'.

If no `which.dataset` is provided upon the call, it can be typed in or selected from the list of existing datasets.

Please note that 'pokeDataset' is also invoked when a new dataset is loaded into memory using the functions 'loadData' or 'accessVar'. In the former case it is stored under the name of the file, in the latter under the variable name. If such a name already exists in 'WRCube', a time stamp is attached.

For restoring the stored variables serve functions 'peekDataset' and 'selectDataset'. The function 'purgeDatasets' removes all older datasets, apart from the most recent copy of the current one.

Value

None.

Warning

If not called from a GUI, no warning is issued upon rewriting the existing dataset.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'peekDataset' 'selectDataset' 'purgeDatasets' 'loadData' 'accessVar'

Examples

```
data(sazava)
accessVar("sazava")
# stored as sazava in WRCube
assignColVar("MgO","blues")
assign1symb(15)
# store a new copy in the WRCube
pokeDataset("coloured sazava",overwrite.warn=FALSE)

data(swiss)
accessVar("swiss")
# stored as swiss in WRCube

peekDataset("sazava")
binary("SiO2","Ba")

peekDataset("coloured sazava")
binary("SiO2","Ba")

peekDataset("swiss")
binary("Catholic","Education",pch=15,col="darkgreen")
```

---

ppm2oxide *Calculation of wt% of the given oxide from ppm of atom*

Description

Recasts concentrations of a cation (in ppm) to those of the selected oxide (in wt %).

Usage

```
ppm2oxide(formula,where="WR")
```

Arguments

formula	character: the oxide which is to be recalculated
where	character: a name of matrix or dataframe with the data to be recalculated

Value

A numeric matrix with one column containing the recalculated concentrations of the given oxide (in wt %) for individual samples.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

oxide2ppm, oxide2oxide, molecularWeight

Examples

```
data(sazava)
accessVar("sazava")

print(WR[,c("K2O","K")])
ppm2oxide("K2O") # Calculates K2O (in wt %) from K (in ppm)

oxide2ppm("K2O") # And vice versa
```

---

prComp *Statistics: Principal components*

Description

Performs principal components analysis (scaled variables, covariance or correlation matrix) and plots a biplot *(Gabriel, 1971)*.

Usage

```
prComp(comp.data=NULL,use.cov=FALSE,scale=TRUE,GUI=FALSE)
```

Arguments

comp.data	a numerical matrix; the data to be normalized. Or just names of variables in the data matrix 'WR'.
use.cov	logical; should be the covariance matrix used instead of correlation matrix?
scale	logical; the scalings applied to each variable.
GUI	logical; is the function called from a menu (GUI)?

Details

Biplot aims to represent both the observations and variables of a data matrix on a single bivariate plot *(Gabriel, 1971; Buccianti & Peccerillo, 1999)*.

In the biplots, the length of the individual arrows is proportional to the relative variation of each variable. A comparable direction of two arrows implies that both variables are positively correlated; the opposite one indicates a strong negative correlation. When two links are perpendicular it indicates independence of the two variables *(Buccianti & Peccerillo, 1999)*.

If called from menu (GUI version), a list of major elements (SiO2, TiO2, Al2O3, FeOt, MnO, MgO, CaO, Na2O, K2O) is assumed as a default, but different variables can be specified by the function 'selectColumnsLabels'.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSamples for details.

Value

Vector of the scores of the supplied data on the principal components is stored in a variable 'results'. Returns invisibly the complete output from the underlying function 'princomp'.

Warning

Names of existing numeric data columns and not formulae involving these can be handled at this stage. Only complete cases are used for the principal components analysis.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Buccianti A & Peccerillo A (1999) The complex nature of potassic and ultrapotassic magmatism in Central-Southern Italy: a multivariate analysis of major element data. In: Lippard S J, Naess A, Sinding-Larsen R (eds) Proceedings of the 5th Annual Conference of the International Association for Mathematical Geology. Tapir, Trondheim, p. 145-150

Gabriel KR (1971) The biplot graphical display of matrices with application to principal component analysis. Biometrika 58: 453-467

See Also

The compositional data should be first transformed to centred-log-ratios (clr) using the function 'clr.trans'. See example. Or use the convenience function pr.comp.clr.

See *Reimann et al. (2008)* with *van den Boogaart and Tolosana-Delgado (2013)* for further details and *van den Boogaart and Tolosana-Delgado (2008)* for implementation of a comprehensive R library dealing with compositional data.

For further info on the used principal components algorithm and biplots, see the R manual entries of 'princomp' and 'biplot.princomp'.

Examples

```
data(sazava)
accessVar("sazava")

ox<-c("SiO2","Al2O3","FeOt","MgO","CaO")
clr.trans(ox)
prComp(results)
```

---

printSamples *Display samples*

Description

Displays specified combination of numeric variable(s) and/or labels for selected range of samples.

Usage

```
printSamples(elems=NULL,which=NULL,select.samples=FALSE,print=TRUE)
```

Arguments

elems	list of variables to be printed
which	list of samples, useful only for select.samples=FALSE
select.samples	logical: if TRUE, samples can be chosen using the appropriate dialogue
print	logical: should be the result indeed printed or just returned for further evaluation?

Details

This function prints the desired numerical columns, textual labels, or their combinations, for selected samples.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSamples for details.

The variables to be printed are chosen by the function 'selectColumnsLabels'. In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

results	data matrix with the desired data for the specified samples

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
## Not run:
# Querying names of numeric data columns

Search pattern = SiO2, MgO, CaO

Search pattern = major
SiO2, TiO2, Al2O3, Fe2O3, FeO, MnO, MgO, CaO, Na2O, K2O, P2O5

Search pattern = LILE
Rb, Sr, Ba, K, Cs, Li

Search pattern = HFSE
Nb, Zr, Hf, Ti, Ta, La, Ce, Y, Ga, Sc, Th, U

Search pattern = REE
La, Ce, Pr, Nd, Sm, Eu, Gd, Tb, Dy, Ho, Er, Tm, Yb, Lu

Search pattern = Locality,SiO2,LILE,HFSE
Locality, SiO2, Rb, Sr, Ba, K, Cs, Li, Nb, Zr, Hf, Ti,
Ta, La, Ce, Y, Ga, Sc, Th, U

Search pattern = 1:5, 7
Numeric data columns number 1, 2, ...5, 7

# User-defined list
my.elems<-c("Rb","Sr","Ba")
Search pattern = my.elems
Rb, Sr, Ba
```

```
## End(Not run)
```

---

printSingle *Display a variable*

Description

Displays a single numeric variable or a result of a calculation.

Usage

```
printSingle(default="")
```

Arguments

default	character: list of default column names, separated by commas.

Details

The variable to be printed is selected using the function 'selectColumnLabel'. In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

results	numerical vector/matrix with the results

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
## Not run:
# examples of valid formulae....
(Na2O+K2O)/CaO
Rb^2
log10(Sr)
mean(SiO2)/10

# ... but this command is in fact a simple R shell -
# meaning lots of fun for power users!
summary(Rb,na.rm=TRUE)
cbind(SiO2/2,TiO2,Na2O+K2O)
cbind(major)
hist(SiO2,col="red")
boxplot(Rb~factor(groups))

# possibilities are endless
plot(Rb,Sr,col="blue",pch="+",xlab="Rb (ppm)",ylab="Sr (ppm)",log="xy")

## End(Not run)
```

`profiler` *Profile plotting*

Description

Plotting geochemical profiles. As a x axis can be specified an arbitrary variable or an numerical interval (for equidistant measurements).

Usage

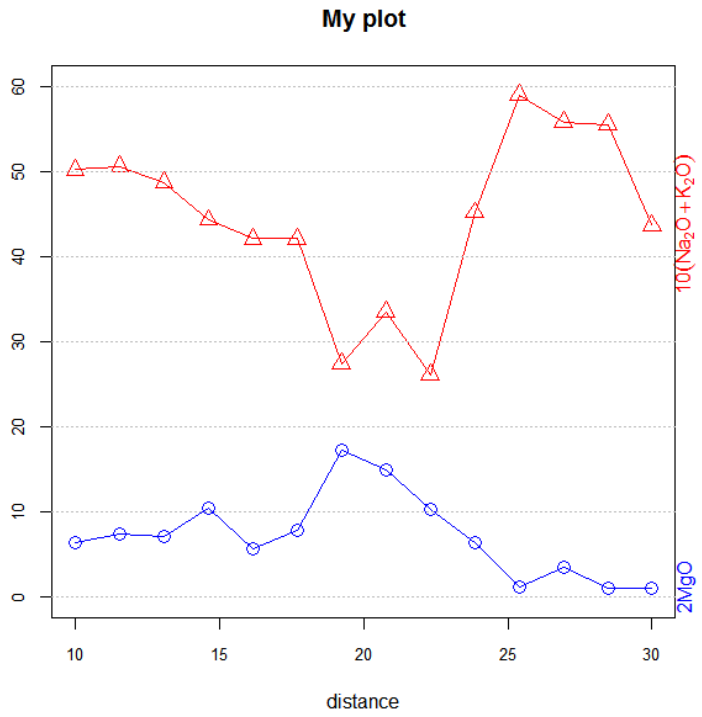
```
profiler(x = NULL, y = NULL, data = WR, method = "Variable", legend = FALSE,
    pch = 1, col.lines = "black", col = "black", cex = 1,
    xaxs = "r", yaxs = "i",
    main = "",xmin = NULL, xmax = NULL, ...)
```

Arguments

`x`	character; optional name of variable to be plotted as x axis.
`y`	character; name(s) of variable(s) for individual profiles.
`data`	numeric; a matrix with the data.
`method`	character; which of the methods is to be used? Valid are "Variable","Equidistant" or "From-To".
`legend`	logical; should be plotted also legend (in a separate window)?
`pch`	plotting symbols specification.
`col.lines`	colour(s) for connecting lines.
`col`	plotting colour(s).
`cex`	numeric; relative size of the plotting symbols.
`xaxs, yaxs`	character; type of the axes. See `par` for details.
`main`	character; main title for the plot
`xmin, xmax`	range of the x axis (for methods '`Variable`' and '`From/To`'))
`...`	any additional parameters for the function `lines`.

Details

The function '`profiler`' serves for plotting three different types of profiles involving a single or several geochemical parameters.



The first one, '`Variable`' uses any numeric variable as the x axis (e.g., SiO2 contents, depth...). It is in fact a special type of a binary plot, in which the data points are, for each of the y-axis variables, joined by a line.

The remaining two methods are very similar to each other. The x axis is in both cases equidistant, and the order of the individual samples follows from their sequence in the data set.

The method '`Equidistant`' uses simply the sequence number of the individual samples in the data set. It does not label the x-axis, just prints the number of samples used for plotting.

The method '`From/To`' serves for drawing equidistant profiles, where the x axis can be specified by an interval.

In the specification of the x axis (for the method '`Variable`') or any of the y variables (all methods) can be used also arithmetic expressions, see `calcCore` for the correct syntax.

If not called from the command prompt, the samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSubset` for details.

The easiest way to specify the variable(s) to be plotted on individual profile(s) is to type directly the names of the columns, separated by commas. Alternatively can be used their sequence numbers or ranges. Also built-in lists can be employed, such as 'LILE', 'REE', 'major' and 'HFSE' or their combinations with the column names.

These lists are simple character vectors, and additional ones can be built by the user (see Examples). Note that currently only a single, stand-alone, user-defined list can be employed as a search criterion.

If the function is not called from the command prompt, and it desired so, the symbols and colours for each of the profiles can be specified separately in a simple spreadsheet-like interface.

If x axis occurs among the arguments to be plotted as y axes, it is skipped.

Likewise the relative scaling of the plotting symbols and the scale of the y axis can be specified.

Lastly, the user is asked to enter the limits for the axes, which are always two numbers separated by a comma.

Value

`results` numeric matrix with the values for individual profiles.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(sazava)
accessVar("sazava")

# Profiles of SiO2 versus (scaled) TiO2, MgO and K2O
# if x is specified, method="Variable" is assumed automatically
profiler("Na2O+K2O",c("TiO2","6*MgO","SiO2"), pch = c("+","o","@"),
    col = c("red","blue","darkgreen"),
    col.lines = c("red","blue","darkgreen"),
    xmin = 2, xmax = 6, legend=TRUE)

# Equidistant profiles of (scaled) MgO, CaO, and Al2O3 (in sample sequence)
# with specified symbols, scaling and line styles
profiler(y = c("MgO","3*CaO","2*Al2O3"), method="Equidistant",
    col = "black", pch = 15, cex = 1.5,
    col.lines = c("red","blue","darkgreen"), legend = TRUE, lwd = 3)

# just lines
profiler(y = c("MgO","3*CaO","2*Al2O3"), method="Equidistant",
    col = "transparent", col.lines = c("red","blue","darkgreen"),
    legend = TRUE, lwd = 3)

# Equidistant profiles of two calculated variables in custom colour
# and user-defined plotting symbols; range of the x axis will be specified
# automatically
profiler(y = c("2*MgO","10*(Na2O+K2O)"), method = "From-To", pch = 1:2,
  col = c("blue","red"), cex = 1.5, main = "My plot", xmin = 10,xmax = 30,legend = TRUE)
```

---

psAll *Save all graphics to PS*

Description

Saves all graphical windows to Postscript files.

Usage

```
psAll(filename=NULL)
```

Arguments

`filename` a name of file for saving the output.

Details

The function prompts for a common root of the filenames and then saves all graphical windows, each in a separate file, numbering them sequentially. Postscript is the best export format from R, preserving the necessary quality as well as the possibility to be imported by most graphical editors (such as Corel Draw!) for retouching.

Otherwise individual diagram can be saved from a menu that appears after clicking on the appropriate graphical window ('File|Save as|Postscript').

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'pdfAll' 'postscript'

---

purgeDatasets *Removing stored datasets from the memory*

Description

Removes all the stored datasets (apart from the current one) in order to save memory.

Usage

```
purgeDatasets(GUI=FALSE)
```

Arguments

GUI	logical; is the function called from GUI?

Details

This function removes all older datasets, regardless whether stored automatically by the functions 'loadData' or 'accessVar', as well as on demand by 'pokeDataset'.

Only the most recent copy of the current dataset is preserved (i.e. the last item within the list 'WRCube').

Value

None.

Warning

If not called from a GUI, no warning is issued and all but the current dataset are deleted immediately.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'pokeDataset' 'peekDataset' 'selectDataset'

---

pyroxene *Selected pyroxene analyses*

Description

This data set gives the selected chemical analyses of pyroxene from *Deer et al. (2013)*.

Usage

```
data(pyroxene)
```

Format

A data frame containing 15 analyses.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Deer WA, Howie RA, Zussman J (2013) An Introduction to the Rock-Forming Minerals. Mineralogical Society, London, p 98 doi: 10.1180/DHZ

See Also

data, accessVar

Examples

```
data(pyroxene)
accessVar("pyroxene")
print(WR)
```

quitGCDkit *Exit GCDkit*

Description

Exits GCDkit (nicely).

Usage

```
quitGCDkit()
```

Arguments

None.

Details

By invoking this command the user is not prompted whether he wants to save his unfinished work in the 'Workspace image', i.e. file '.RData' in the main GCDkit directory.

Menu

GCDkit: Exit GCDkit

See Also

'quit'

r2clipboard *Copy results to clipboard*

Description

Copies the most recently calculated results to a clipboard.

Usage

```
r2clipboard(what=results)
```

Arguments

what	a variable to be copied, can be either a vector, a matrix, a list or a table.

Details

Copies the variable 'results' returned by most of the calculation algorithms to the Windows clipboard.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

`recalcOptions` *Access the recalculation database*

Description

The function gathers information on recalculation options for the given mineral and prepares it for printing.

Usage

```
recalcOptions(mineral=NULL,warning=TRUE)
```

Arguments

`mineral`	character; name of the mineral class
`warning`	logical; should be a warning shown when mineral is not found?

Details

This function prints all recalculation options available in the database (i.e., the file `'mineral_db.r'`) for the given mineral class.

If no mineral is specified, the user has a choice from a drop-down list of the minerals present in the current dataset.

Value

An object of the class `'mineral.db'` (essentially a list, for which there is implemented a new method to the generic function `'print'`).

Author(s)

Vojtech Janousek, <vojtech.janousek@geology.cz>

See Also

`mineral-class`

Examples

```
data(combined)
# Automatic default recalculation, as specified in the database
accessVar(combined)
recalcOptions("garnet")
# a mineral from the current version of the database, formatted nicely
```

recast *Recast to given sum*

Description

Recalculation of the selected data to a fixed sum.

Usage

```
recast(total = 100)

normalize2total(what = NULL, total = 100)
```

Arguments

what	numeric matrix or character vector with a list of column names to be normalized, separated by commas.
total	a sum the data should be normalized to.

Details

Both functions return the selected elements/oxides (by default columns in the data matrix 'WR') normalized to the required sum. The function 'recast' is front-end to 'normalize2total'. If 'what' is a comma delimited list, the corresponding columns from the data matrix 'WR' are selected. If 'what' is empty, the user is prompted to supply the list of required column names via the function 'selectColumnsLabels'.

Value

results	numerical vector/matrix with the results

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(sazava)
accessVar("sazava")

normalize2total(major,1)
# Use recast() to select the sum and elements interactively
```

`Regular expressions` *Implementation of regular expressions in GCDkit*

Description

Implementation of regular expressions in the searching patterns.

Details

Many enquiries in the GCDkit employ regular expressions. This is a quite powerful searching mechanism more familiar to people working in Unix. Put in simple terms, most characters, including all letters and digits, are regular expressions that match themselves. However, metacharacters with a special meaning ('?' '+' '{' '}' '|' '(' ')') must be preceded by a backslash.

Regular expression	Matches
.	Any character
^	Beginning of the expression
\$	End of the expression
[ ]	Any of the characters given in square brackets
[m-n]	Any character in the range given by `m` and `n`

A subexpression is a regular expression enclosed in '\(' and '\)'. Two such subexpressions may be joined by the infix operator '|' (logical or); the resulting regular expression matches any string matching either of them. For instance:

```
\(South\)|\(North\)Uist
```

yields both

`South Uist` and `North Uist`.

A regular expression may be followed by one of several repetition operators:

Repetition operator	The preceding item will be matched
?	At most once (i.e. is optional)
*	Zero or more times
+	One or more times
{n}	Exactly `n` times
{n,}	At least `n` times
{n,m}	At least `n` times, but not more than `m` times

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

`regex`

Examples

```
## Not run:
```

```
# Subset by label
The searched field corresponds to localities with the following levels:
Mull, Rum, Skye, Coll, Colonsay, Hoy, Westray, Sanday,
Stronsay, Tiree, Islay

Search pattern = ol
Coll, Colonsay

Search pattern = n.a
Colonsay, Sanday, Stronsay

Search pattern = ^S
Skye, Sanday, Stronsay

Search pattern = e$
Skye, Tiree

Search pattern = [ds]ay
Colonsay, Sanday, Stronsay

Search pattern = [p-s]ay
Colonsay, Westray, Stronsay

Search pattern = ol|oy
Coll, Colonsay, Hoy

Search pattern = l{2}
Mull, Coll


# Subset by sample name
The sample names are: Bl-1, Bl-3, Koz-1, Koz-2, Koz-5, Koz-11,
KozD-1, Ri-1.

Search pattern = oz-[1-3]
Koz-1, Koz-2, Koz-11

Search pattern = oz-|Bl-
Bl-1, Bl-2, Bl-3, Koz-1, Koz-2, Koz-5, Koz-11

## End(Not run)
```

`sampleDataset` *Sample datasets*

Description

Loads a recalculates one of the sample datasets stored as a *.CSV file.

Usage

```
sampleDataset(mineral = NULL)
```

Arguments

`mineral`	character; name of the dataset

Value

WR	numeric matrix: all numeric data
labels	data frame: all at least partly character fields; labels$Symbol contains plotting symbols and labels$Colour the plotting colours

The function prints a short summary about the attached data. It also loads and executes the Plugins, i.e. all the R code that is currently stored in the subdirectory '\Plugin'.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

accessVar

Examples

```
sampleDataset("atacazo")
print(WR)
```

---

saveData *Save data file*

Description

Saves modified data set into a specified datafile.

Usage

```
saveData(sep="\t")
```

Arguments

sep	delimiter separating individual items in the data file.

Details

Labels (stored in data frame 'labels') and numeric data (in numeric matrix 'WR') for the currently selected subset are glued together and saved under the specified filename. The format is such that the data can be retrieved again into GCDkit using the loadData command. Note that no mg numbers are currently saved.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'loadData' 'mergeData' 'showColours' 'colours' 'showSymbols' 'read.table'

saveResults *Save results*

Description

Saves the most recently calculated results to a text file.

Usage

```
saveResults(what = results, sep = "\t", digits = 2)
```

Arguments

what	a variable to be saved, can be either a vector, a matrix or a list.
sep	separator; default is a tab-delimited file.
digits	precision of the results to be saved.

Details

Saves the variable 'results' returned by most of the calculation algorithms to a tab-delimited ASCII file.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

sazava *Whole-rock composition of the Sazava suite, Central Bohemian Plutonic Complex*

Description

This data set gives the whole-rock major- and trace-element contents in selected samples (gabbros, quartz diorites, tonalites and trondhjemites) of the c. 355 My old calc-alkaline Sazava suite of the Variscan Central Bohemian Plutonic Complex (Bohemian Massif, Czech Republic).

Usage

```
data(sazava)
```

Format

A data frame containing 14 observations.

Source

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Janoušek V, Rogers G, Bowes DR (1995) Sr-Nd isotopic constraints on the petrogenesis of the Central Bohemian Pluton, Czech Republic. Geol Rundsch 84: 520-534 doi: 10.1007/BF00284518

Janoušek V, Bowes DR, Rogers G, Farrow CM, Jelínek E (2000) Modelling diverse processes in the petrogenesis of a composite batholith: the Central Bohemian Pluton, Central European Hercynides. J Petrol 41: 511-543 doi: 10.1093/petrology/41.4.511

Janoušek V, Braithwaite CJR, Bowes DR, Gerdes A (2004) Magma-mixing in the genesis of Hercynian calc-alkaline granitoids: an integrated petrographic and geochemical study of the Sazava intrusion, Central Bohemian Pluton, Czech Republic. Lithos 78: 67-99 doi: 10.1016/j.lithos.2004.04.046

Examples

```
data(sazava)
accessVar("sazava")
binary("SiO2","Ba")
```

`scattersmooth` *scattersmooth*

Description

Plotting scatterplot with smoothed densities (smoothed two-dimensional histogram) after *Eilers & Goeman (2004)*.

Usage

```
scattersmooth(xlab = NULL, ylab = NULL, samples = NULL, nbin = 100, lambda = 1,
    pal = heat.colors(100), pch = 15, col = "blue", cex = 0.3,
    xlim = NULL, ylim = NULL, ...)
```

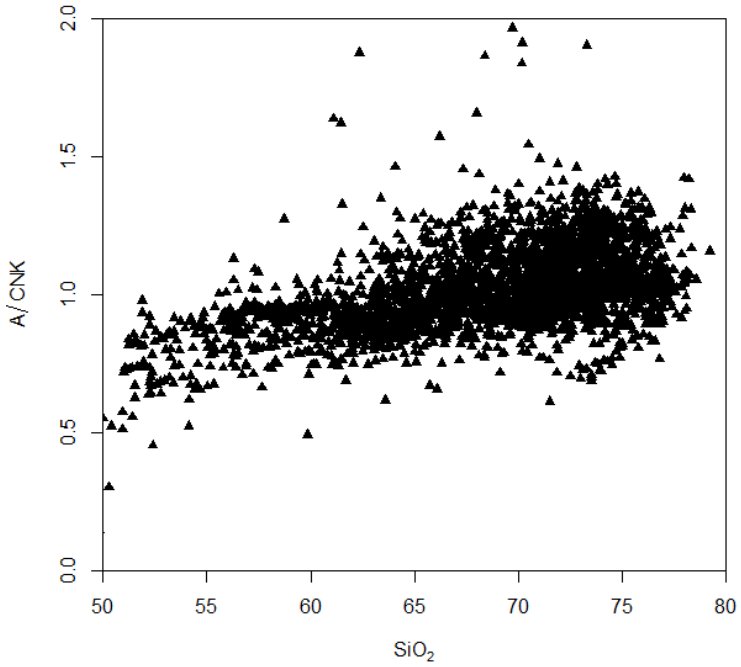
Arguments

`xlab, ylab`	character; specification of the plotting variables (formulae OK).
`samples`	character or numeric vector; specification of the samples to be plotted.
`nbin`	integer, giving the number of bins for x and y, or a vector with two integers.
`lambda`	the smoothing parameter; larger lambda gives smoother curves.
`pal`	specification of a palette yielding colours for filled contours.
`pch`	plotting symbols.
`col`	plotting colours.
`cex`	relative size of plotting symbols.
`xlim`	limits of the x axis.
`ylim`	limits of the y axis.
`...`	Further parameters to the original function.

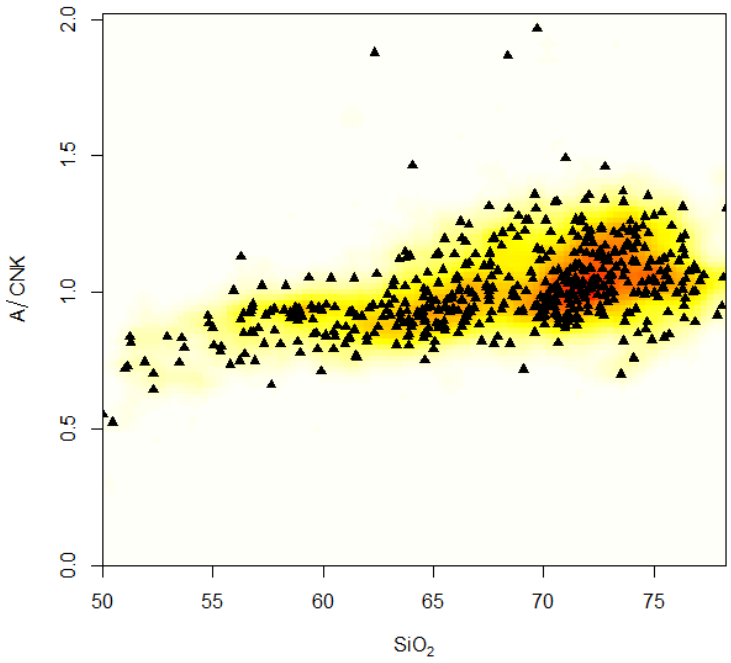
Details

The function produces a scatterplot with smoothed densities (smoothed two-dimensional histogram). The code has been adopted, with only small modifications and new interface to GCDkit, from the original R functions designed by *Eilers & Goeman (2004)*.The original scatterplot function has been renamed to '`.scattersmoothMain`'.

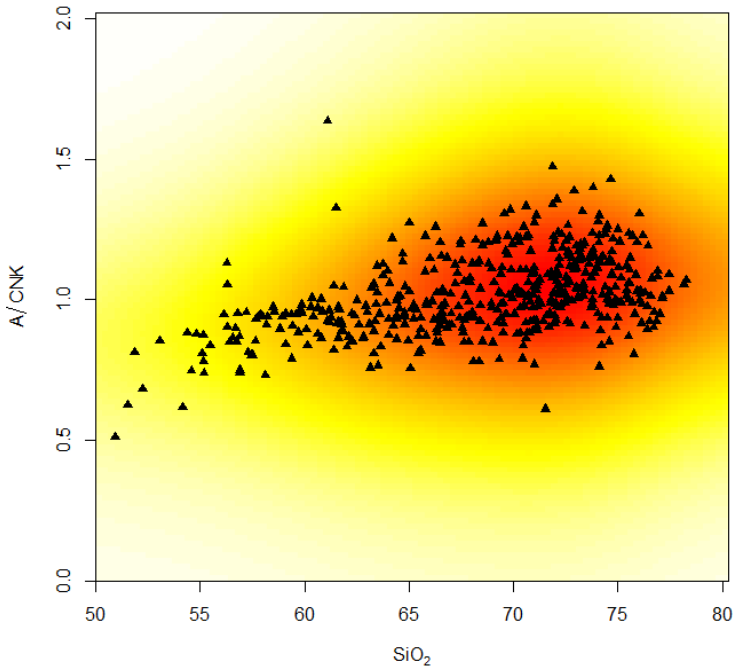
The complete dataset:



If plotted with `lambda=1`:



If plotted with `lambda=100`:



The variables to be plotted are selected using the function '`selectColumnLabel`'. In the specification of the variables can be used also arithmetic expressions, see `calcCore` for the correct syntax.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSubset` for details.

Value

Returns (invisibly) all parameters calculated by the original function.

Warning

The function is NOT Figaro-compatible.

Author(s)

Paul H. C Eilers, <p.eilers@erasmusmc.nl>

& Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Eilers PHC, Goeman JJ (2004) Enhancing scatterplots with smoothed densities. Bioinformatics 20: 623-628

Examples

```
data(atacazo)
accessVar("atacazo")

scattersmooth("SiO2","Na2O+K2O",pal=heat.colors(100),lambda=10)

scattersmooth("SiO2","Na2O+K2O",pal=heat.colors(100),pch=17,
    col="black",cex=1,lambda=100)
```

---

selectAll *Select whole dataset*

Description

Restores data for all samples as they were loaded from a data file.

Usage

```
selectAll(GUI=FALSE)
```

Arguments

GUI	logical; was the function called from the GUI?.

Details

When a datafile is loaded into GCDkit using the loadData function, the data and their backup copy are stored in the memory.

The subsets of the current dataset can be chosen using the functions selectByLabel and selectSubset (menus 'Select subset by sample name or label', 'Select subset by range' , 'Select subset by Boolean') and the current data will be replaced by their newly chosen subset.

The backup copy is kept intact ever since the loadData function has been invoked and can be uploaded any time in place of the current data set using the function 'selectAll'. Note that all changes made e.g. to plotting symbols, grouping, newly calculated variables etc. will be lost.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

---

selectByLabel *Select subset by sample name or label*

Description

Selecting subsets of the data stored in memory by searching sample names or a single label.

Usage

```
selectByLabel()
```

Details

This function enables the user to query a single textual column, a label, chosen using the function 'selectColumnLabel'. The current data will be replaced by its newly chosen subset. These enquiries employ regular expressions.

Value

Overwrites the data frame 'labels' and numeric matrix 'WR' by subset that fulfills the search criteria.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
## Not run:
# Subset by label
The searched field corresponds to localities with the following levels:
Mull, Rum, Skye, Coll, Colonsay, Hoy, Westray,
Sanday, Stronsay, Tiree, Islay

Search pattern = ol
Coll, Colonsay

Search pattern = n.a
Colonsay, Sanday, Stronsay

Search pattern = ^S
Skye, Sanday, Stronsay

Search pattern = e$
Skye, Tiree

Search pattern = [ds]ay
Colonsay, Sanday, Stronsay

Search pattern = [p-s]ay
Colonsay, Westray, Stronsay

Search pattern = ol|oy
Coll, Colonsay, Hoy

Search pattern = l{2}
Mull, Coll


# Subset by sample name
The sample names are: Bl-1, Bl-3, Koz-1, Koz-2,
Koz-5, Koz-11, KozD-1, Ri-1.

Search pattern = oz-[1-3]
Koz-1, Koz-2, Koz-11

Search pattern = oz-|Bl-
Bl-1, Bl-2, Bl-3, Koz-1, Koz-2, Koz-5, Koz-11

## End(Not run)
```

`selectByMineral` *Return a slot for the selected mineral class(es)*

Description

Returns data stored in the given slot for selected mineral(s)

Usage

```
selectByMineral(slot=NULL,whichmins=NULL,multiple=TRUE,digits=4,silent=FALSE)
```

Arguments

slot	character: name of the desired slot
whichmins	charater: which minerals are to be selected?
multiple	logical: is a multiple choice of minerals feasible?
digits	numeric: how many decimal places are to be reported?
silent	logical: should be the result printed?

Details

In GUI, the desired mineral(s) is to be chosen from a drop down list of available mineral classes, as is a (single) slot if 'slot=NULL'.

Value

A list, with components for each mineral class, with the desired data.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(combined)
accessVar("combined")
selectByMineral(slot="rough",whichmins=c("garnet","feldspar"))

selectByMineral(slot="formula",whichmins="garnet",digits=5)
```

---

selectColumnLabel *Selecting a single variable in GCDkit*

Description

This is an auxiliary function invoked by many others to select a single variable.

Usage

```
selectColumnLabel(where = colnames(labels),
message = "Select the variable\nor press ENTER to pick from a list",
default = "", sample.names = FALSE, silent = FALSE, print = TRUE,
empty.ok = TRUE)
```

Arguments

where	names of data columns to choose from
message	prompt
default	comma delimited list of default names
sample.names	logical; should be the sample names listed
silent	logical, echo on/off
print	logical, echo on/off
empty.ok	is empty selection ok?

Details

The easiest way for specification of the variable is to type directly the name of the numerical column in the data matrix 'WR' (e.g., 'SiO2') or its sequence number (2 for the second column). However, it is not necessary to enter the name in its entirety. Only a substring that appears somewhere in the column name or other forms of regular.expressions can be specified.

If the result is ambiguous, the correct variable has to be selected by mouse from the list of the multiple matches. Ultimately, empty response invokes list of all variables available in the memory.

Value

A numeric index of the selected column.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

selectColumnsLabels

---

selectColumnsLabels *Selecting several data columns*

Description

An auxiliary function invoked by many others to select several variables simultaneously.

Usage

```
selectColumnsLabels(where = colnames(WR),
message = "Select variable(s), e.g. 'SiO2,TiO2,MgO'
or press ENTER to pick from a list", default = "", print = TRUE,
exact.only = TRUE)
```

Arguments

where	vector of names for data columns to choose from
message	prompt
default	comma delimited list of default names
print	logical, echo on/off
exact.only	logical, should be the input checked for correctness?

Details

The variable(s) can be specified in several ways. The easiest is to type directly the name(s) of the column(s), separated by commas. Alternatively can be used their sequence numbers or ranges. Also built-in lists can be employed, such as `'LILE'`,`'REE'`,`'major'` and `'HFSE'` or their combinations with the column names.

These lists are simple character vectors, and additional ones can be built by the user (see Examples). Note that currently only a single, stand-alone, user-defined list can be employed as a search criterion.

Empty response invokes list of all variables available. The correct variables have to be selected by mouse + SHIFT from this list.

If `exact.only=TRUE`, the individual items in the input line are checked against the list of existing column/variable names (i.e. components in the vector `'where'`).

Value

Vector with the selected column names.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
## Not run:
# Querying names of numeric data columns

Search pattern = SiO2, MgO, CaO

Search pattern = major
SiO2, TiO2, Al2O3, Fe2O3, FeO, MnO, MgO, CaO, Na2O, K2O, P2O5

Search pattern = LILE
Rb, Sr, Ba, K, Cs, Li

Search pattern = HFSE
Nb, Zr, Hf, Ti, Ta, La, Ce, Y, Ga, Sc, Th, U

Search pattern = REE
La, Ce, Pr, Nd, Sm, Eu, Gd, Tb, Dy, Ho, Er, Tm, Yb, Lu

Search pattern = Locality,SiO2,LILE,HFSE
Locality, SiO2, Rb, Sr, Ba, K, Cs, Li, Nb,
Zr, Hf, Ti, Ta, La, Ce, Y, Ga, Sc, Th, U

Search pattern = 1:5, 7
Numeric data columns number 1, 2, ...5, 7

# User-defined list
my.elems<-c("Rb","Sr","Ba")
Search pattern = my.elems
Rb, Sr, Ba

## End(Not run)
```

selectPalette *selectPalette*

Description

Picks given number of colour shades from one of the available palettes.

Usage

```
selectPalette(n,colour.palette=NULL,GUI=TRUE)
```

Arguments

n	desired number of colours
colour.palette	one of the colour palette names, see Details
GUI	logical; is the function called from GUI?

Details

The desired number of colours has to be given in any case.

The possible palettes are: 'grays','reds','blues','greens','cyans','violets','yellows', 'cm.colors','heat.colors','terrain.colors','topo.colors','rainbow' and 'jet.colors'.

Also, user-defined palette functions are supported. See Examples.

If not specified upon function call, the colour palette can be picked from list of available ones. Optionally (if GUI = TRUE) a chart with their preview is shown.

Value

Returns a matrix with a single row of hexadecimal codes. Its single value of `rownames` represents the name of the palette selected.

Note

Note that UK spelling of "colours" in names of palettes is fixed automatically to the US "colors".

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

Table of the available named plotting colours is obtained by `showColours`.

assignColLab assignColVar plotWithCircles filledContourFig

Examples

```
selectPalette(5,"heat.colours")

my.palette<-colorRampPalette(c("black", "darkgreen", "red"),space = "rgb")
selectPalette(5,"my.palette")
```

`selectSubset` *Select subset*

Description

Selects samples corresponding to given criteria.

Usage

```
selectSubset(what = NULL, where = cbind(labels,WR), save = TRUE, multiple = TRUE,
    text = "Press ENTER for all samples, or specify search pattern
 by sample name, range or Boolean condition",
    range = FALSE, GUI = FALSE, all.nomatch = TRUE)

selectSamples(what = NULL, print = TRUE, multiple = TRUE, text = NULL)
```

Arguments

`what`	search pattern
`where`	data to be searched
`save`	should the newly selected subset replace the data in memory, i.e. '`labels`' and '`WR`
`multiple`	logical, can be multiple items selected?
`text`	text prompt
`range`	logical: is the search pattern to be interpreted as a range of samples?
`GUI`	logical: is the function called from within GUI?
`all.nomatch`	logical: return all samples when there is no match?
`print`	logical: should be the chosen samples ID printed?

Details

The function '`selectSubset`' has two purposes.

1. If '`save=TRUE`', it is a core function used in selecting subsets of the current data set by ranges (see `subsetRange`) or Boolean conditions (see `subsetBoolean`).

2. If `save=FALSE`, no permanent subsetting takes place. This is useful for temporary selections of the data, e.g. in determining which samples are to be plotted on a diagram.

In this case, the samples can be selected based on combination of three searching mechanisms. The search pattern is first tested whether it obeys a syntax of a valid `regular expression` that could be interpreted as a query directed to the sample name(s).

If not, the syntax of the search pattern is assumed to correspond to a selection of sample sequence numbers.

At the last resort, the search pattern is interpreted as a Boolean condition that may employ most of the comparison operators common in R, i.e. < (lower than), > (greater than), <= (lower or equal to), >= (greater or equal to), = or == (equal to), != (not equal to). The character strings should be quoted. Regular expressions can be employed to search the textual labels.

The conditions can be combined together by logical `and`,`or` and brackets.

Logical and can be expressed as .and. .AND. &

Logical or can be expressed as .or. .OR. |

The function 'selectSamples' is a front-end to 'selectSubset'.

Value

If 'save=TRUE', the function overwrites the data frame 'labels' and numeric matrix 'WR' by subset that fulfills the search criteria. Otherwise names of samples fulfilling the given criteria are returned.

Warning

So far only names of existing numeric data columns and not formulae involving these can be handled.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

regex, selectByLabel and selectAll

Examples

```
data(sazava)
accessVar("sazava")

# permanent selection, the variables 'WR' and 'labels' affected
selectSubset("SiO2>70")

# back to the complete, originally loaded dataset
selectAll()

# both expressions below return only sample names of analyses fulfilling
# the given criteria, variables 'WR' and 'labels' NOT affected
selectSamples("SiO2<70&MgO>5")

selectAll()
selectSubset("SiO2<70&MgO>5",save=FALSE)
print(WR)

# This one is a permanent selection based on a Boolean condition
# Note the use of backslash as an escape character for quotation marks
selectAll()
selectSubset("Intrusion=\"Pozary\"&SiO2>70",save=TRUE)
print(WR)

## Not run:
#EXAMPLES OF SEARCHING PATTERNS
# Searching by sample name

The sample names are: Bl-1, Bl-3, Koz-1, Koz-2,
Koz-5, Koz-11, KozD-1, Ri-1.

oz-[1-3]
# Samples Koz-1, Koz-2, Koz-11
```

```
oz-|Bl-
# Samples Bl-1, Bl-2, Bl-3, Koz-1, Koz-2, Koz-5, Koz-11

# Searching by range

1:5
# First to fifth samples in the data set

1,10
# First and tenth samples

1:5, 10:11, 25
# Samples number 1,  2, ...5, 10, 11, 25

# Searching by Boolean
######################

Intrusion="Rum"
# Finds all analyses from Rum

Intrusion="Rum".and.SiO2>65
Intrusion="Rum".AND.SiO2>65
Intrusion="Rum"&SiO2>65
# All analyses from Rum with silica greater than 65
# (all three expressions are equivalent)


MgO>10&(Locality="Skye"|Locality="Islay")
# All analyses from Skye or Islay with MgO greater than 10

Locality="^S"
# All analyses from any locality whose name starts with capital S

## End(Not run)
```

`setCex` *Set uniform symbols size*

Description

Defines the default relative size of plotting symbols.

Usage

```
setCex(x)
```

Arguments

`x`	numeric; scaling for the plotting symbols.

Details

The coefficient determining the plotting symbols expansion is stored in a variable '`labels[,"Size"]`', the default is 1.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

gcdOptions

Examples

```
data(sazava)
accessVar("sazava")

setCex(2) # double size
plotDiagram("TAS",FALSE)

setCex(0.5) # half the size
plotDiagram("TAS",FALSE)
```

---

setShutUp *Quiet mode?*

Description

Determines whether extensive textual output is to be printed.

Usage

```
setShutUp()
```

Arguments

None.

Details

The control option is shut.up, whose default is FALSE, meaning that detailed information is to be printed. This, however, may become not viable on slower systems and/or for extensive data sets.

This can be set from the menu 'GCDkit|Options' by setting the checkbox 'Minimize output on screen?' or directly, from the command line (see Examples).

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'gcdOptions' 'options'

Examples

```
getOption("gcd.shut.up")    # query the current value of the given option
options("gcd.shut.up"=TRUE) # reduce the printed output to a minimum
```

`setTransparency` *Setting transparency of plotting colours*

Description

Sets transparency of plotting colours for selected samples. Alternatively, it just returns the hexadecimal code(s) of specified colour(s) with the desired degree of transparency.

Usage

```
setTransparency(which.samples=NULL,transp=NULL,alpha=NULL,
col.in="black",save=TRUE,GUI=FALSE)
```

Arguments

`which.samples`	list of samples; if NULL a dialogue is displayed
`transp`	numeric; transparency to be set
`alpha`	character; alpha value to be set (opacity)
`col.in`	numeric or character vector; colour specification(s)
`save`	logical; should be the result saved into labels$Colour?
`GUI`	logical; is the function called form within GUI?

Details

The transparency value has to fall between 1 (completely transparent) to 0 (opaque).

Alternatively, the so-called alpha channel can be specified, which can attain any hexadecimal number between 0 (completely transparent) to ff (opaque).

if `GUI = TRUE`, the samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see `selectSamples` for details.

Value

Returns (invisibly) hexadecimal codes of the colours with desired degree of transparency If '`save=TRUE`' it also assigns '`labels$Colour`' producing the new, partly transparent colour.

Warning

As a side product, plotting colours are converted to hexadecimal values, which are not easy to translate back to symbolic names.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

Colours by a single variable can be assigned by `assignColLab`, symbols and colours by groups simultaneously by `assignSymbGroup`. Uniform colours are obtained by `assign1col`. Table of available plotting colours is obtained by `showColours`.

Examples

```
data(sazava)
accessVar("sazava")

# Affects the colour of plotting symbols in the system (save=TRUE by default)
ee<-setTransparency(transp=0.5)
binary("SiO2","Na2O+K2O")

setTransparency(transp=0)
setTransparency(which.samples=c("Sa-1","Sa-2","Sa-3"),transp=0.5)
setTransparency(which.samples=c("Sa-1","Sa-2","Sa-3"),alpha="6a")
binary("SiO2","Na2O+K2O")
figCex(2)

# No labels assigned
setTransparency(col=2,transp=0.5,save=FALSE)
setTransparency(col=c("blue","red"),transp=0.5,save=FALSE)
```

---

showColours *Show available colours*

Description

Display colours available for plotting.

Usage

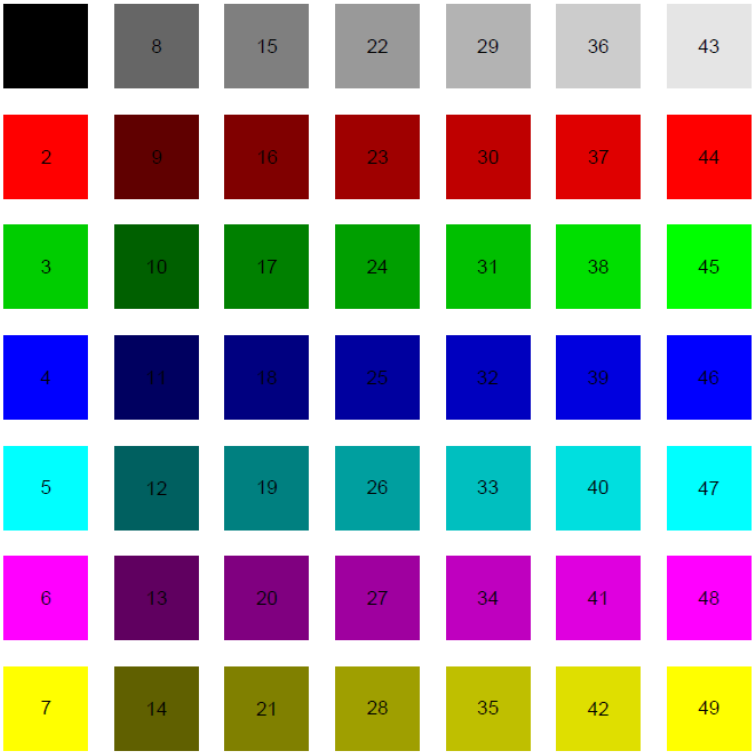
```
showColours(n=49)
showColours2(n=64)
```

Arguments

n	numeric: number of colours to display

Details

The function 'showColours' displays a palette of plotting colours which can be specified by their numeric codes (1-49). On the other hand, 'showColours2' demonstrates the colours which can be given by their English names (there are some 657 of them).



Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

'colours'

---

showLegend *Display legend*

Description

Displays a graphical legend(s) with assignment of plotting symbols and colours used by majority of the diagrams.

Usage

```
showLegend(pch = labels$Symbol, col = labels$Colour, new.plot = TRUE,
   x = "topright", y = NULL, alt.leg = FALSE, just.colours = FALSE, GUI = FALSE, ...)
```

Arguments

pch	numeric or character: plotting symbols.
col	numeric: code for their colour.
new.plot	logical: shall be opened a new plotting window for the legend?

x,y	coordinates for the legend.
alt.leg	logical; should be the alternative (continuous) legend shown? See details.
just.colours	logical; in cases when two legends would be created, should be only that for plotting colours shown?
GUI	logical; Is the function called from GUI (and not batch mode)?
...	any additional parameters for the function legend.

Details

The internal variables 'leg.col' and 'leg.pch' are set to zero, if the current assignment is on the basis of 'groups'. Otherwise they contain the sequential number(s) of column(s) in the data frame 'labels' whose levels are to be used to build the legend(s).

If both variables differ, two legends are created, for plotting symbols and colours separately. This is done unless 'just.colours' is set, when only legend for colours is displayed.

If both variables equal zero, the current grouping information is used.

If a complete colour scale is used for plotting symbols, for instance that created by the assignColVar function, an alternative (continuous) legend can be drawn.

Symbols and colours by a single label can be assigned by functions assignSymbLab and assignColLab respectively, symbols and colours by groups simultaneously by assignSymbGroup. Symbols can be colour-coded according to a variable using the function assignColVar. Uniform symbols are obtained by assign1symb, uniform colours by assign1col. Table of available plotting symbols is displayed by showSymbols and colours by showColours.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

legend figLegend

Examples

```
data(sazava)
accessVar("sazava")
groupsByLabel("Intrusion")

plotDiagram("DebonPQ",FALSE,TRUE)

showLegend(x="topleft",bg="pink",new.plot=FALSE)

showLegend(x="bottomleft",bg="#AAAAAAAA",new.plot=FALSE) # Semitransparent

showLegend(x=10,y=100,bg="khaki",new.plot=FALSE)
```

showSymbols *Show available symbols*

Description

Shows numeric codes of symbols available for plotting:

Usage

```
showSymbols()
```

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

statsByGroup *Statistics by groups*

Description

Calculates simple descriptive statistics for individual columns of the given data matrix; optionally this can be done for each of the groups separately.

Usage

```
statsByGroup(dataset = WR, groups = NULL)
```

Arguments

dataset	numeric data matrix.
groups	a vector, in which is specified, for each sample, a group it belongs to.

Details

The function returns a list containing the calculated statistical parameters respecting the current grouping. The statistical summary involves number of observations, missing values, mean, standard deviation, minimum, 25% quartile, median (= 50% quartile), 75% quartile and maximum. This is a core function invoked both by summarySingle and summarySingleByGroup.

Value

results	a matrix with the means and single standard deviations of both variables for each of the individual groups

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

summarySingle
statistics
summaryAll
summaryByGroup

Examples

```
data(blatna)
accessVar("blatna")
groupsByLabel("Suite")

statsByGroup()

statsByGroup(WR[,LILE])
```

---

statsByGroupPlot *Statistics: Plot summary by element and group*

Description

Plots crosses in a binary diagram denoting means and standard deviations for individual groups.

Usage

```
statsByGroupPlot(xlab,ylab)
```

Arguments

xlab	character; specification of the variable plotted as x axis.
ylab	character; specification of the variable plotted as y axis.

Details

Displays a binary diagram of two elements/oxides in which are plotted averages for the individual groups with whiskers corresponding to their standard deviations.

The variables are entered via the function 'selectColumnLabel'. In the specification of the variables can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

results	a matrix with the results for individual groups and selected two elements/oxides

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(blatna)
accessVar("blatna")
groupsByLabel("Suite")

statsByGroupPlot("SiO2","Na2O/K2O")
```

---

strip *Statistics: Stripplot by groups*

Description

Stripplot for selected samples and variable, respecting the grouping.

Usage

```
strip(xlab = "", ...)
```

Arguments

xlab	variable name
...	additional parameters to stripplot

Details

Stripplot shows 1D scatter plots for each of the groups, with some artificial noise (jitter) added to make the individual points better visible. Stripplots are a good alternative to boxplots when sample sizes are small.

If no variable is specified as 'xlab', the user can enter it using the function 'selectColumnLabel'.

In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

stripplot, stripBoxplot

Examples

```
data(sazava)
accessVar("sazava")
groupsByLabel("Intrusion")

strip("(Na2O+K2O)/Al2O3",cex=2)
```

---

stripBoxplot *Statistics: Stripplot by groups - with boxplots*

Description

Stripplot for selected variable, respecting the grouping. Each of the stripplots for the individual groups are underlain by a boxplot, so that the median, quartiles and range are immediately apparent. Optionally, the data points can be replaced by variously sized/coloured circles, depicting a distribution of a second variable.

Usage

```
stripBoxplot(yaxis="", zaxis="0", ymin=NULL, ymax=NULL, pal="heat.colors",
transp=0, ident=FALSE, scaling.factor=NULL, boxplot.data=NULL, pch=NULL,
col=NULL, cex=NULL, sample.names=FALSE, labs=TRUE, horizontal=FALSE,
silent=TRUE, add=FALSE)
```

Arguments

yaxis	specification of the variable used for stripplots/boxplots.
zaxis	(optional) specification of the variable depicted by the circles.
ymin, ymax	minimum and maximum of the y axis.
pal	name of predefined palette.
transp	numeric, 0-1, transparency of the plotting colours.
ident	logical; should be the samples identified interactively after plotting?
scaling.factor	numeric; relative size of the plotted symbols.
boxplot.data	a list; data for the underlying boxplots (if different from those used for the stripplots). See Details.
pch	plotting symbols.
col	plotting colours.
cex	relative size of the plotting symbols.
sample.names	logical, should be each of the datapoints labeled by a sample name?
labs	logical, should each of the groups labeled by abbreviation of its name?

`horizontal`	logical, should be the plot arranged horizontally?
`silent`	logical, should be some of the above parameters chosen by the appropriate dialogues?
`add`	logical; should be the diagram added to a preexisting plot (rather than a new plotting window opened)?

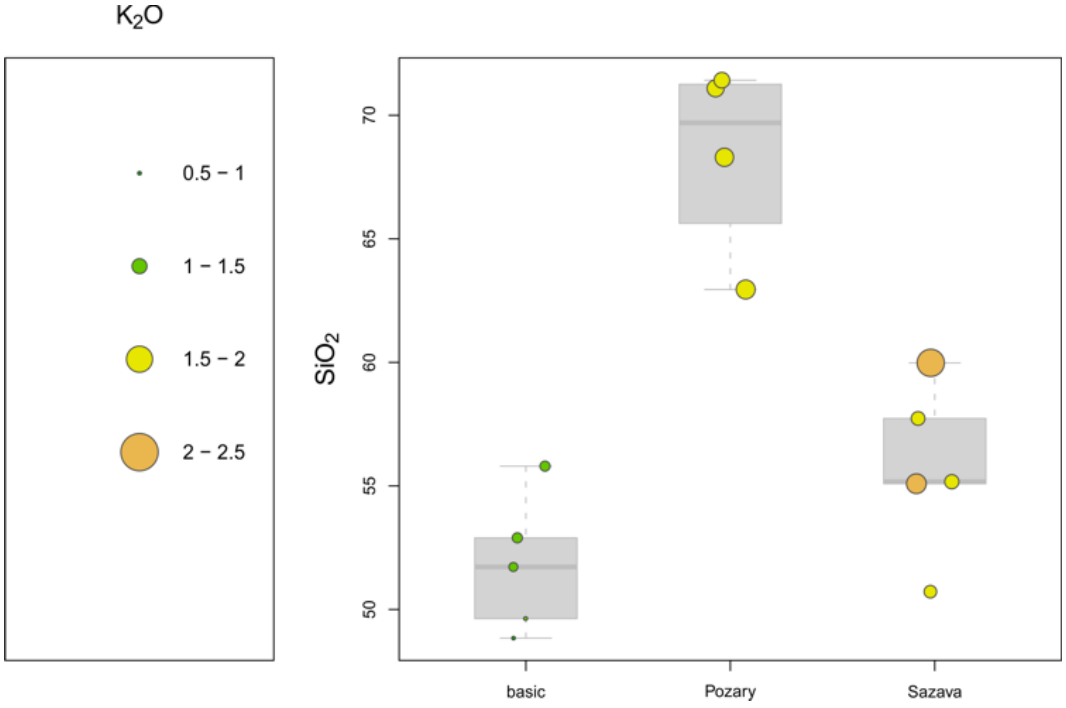
Details

Stripplot shows 1D scatter plots for each of the groups, with some artificial noise (jitter) added to make the individual points better visible. Stripplots are a good alternative to boxplots when sample sizes are small.

If no variable is specified as an argument '`yaxis`', and the function is invoked in interactive regime (`silent = FALSE`), the user can enter it using the function '`selectColumnLabel`'.

If '`zaxis`' is zero, assigned plotting symbols, colours and symbol sizes are used.

If '`zaxis`' refers to a valid variable name, the data points are shown as circles, the size and colours of which correspond to this second variable. In the batch mode, the relative size of the circles plotted can be specified using the parameter `scaling.factor`.



In the specification of the variable(s) can be used also arithmetic expressions, see `calcCore` for the correct syntax.

The colour scheme can be specified by '`pal`'. The legal colour schemes are: `"grays"`, `"reds"`, `"blues"`, `"greens"`, `"cyans"`, `"violets"`, `"yellows"`, `"cm.colors"`, `"heat.colors"`, `"terrain.colors"`, `"topo.colors"`, `"rainbow"` and `"jet.colors"`. Also user-defined palettes are supported, see the Examples.

Normally, the stripplots are underlain by boxplots portraying the statistical distribution of the same data, as used for construction of stripplots for each of the groups. However, with caution, one can specify via `boxplot.data` a list containing the alternative data to be shown on background. Clearly, the number of components in the list, as well as their order, needs to exactly match the individual groups (the levels).

Value

None.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

stripplot, boxplot, strip, plotWithCircles

Examples

```
data(sazava)
accessVar("sazava")
groupsByLabel("Intrusion")

stripBoxplot("(Na2O+K2O)/Al2O3",cex=2)

my.palette<-colorRampPalette(c("black", "darkgreen", "red"),space = "rgb")
stripBoxplot("(Na2O+K2O)/Al2O3","SiO2",pal="my.palette",transp=0.5,ymin=-0.01,ymax=0.5)
```

Subset by range — *Select subset by range*

Description

Selecting subsets of the data stored in memory by their range.

Details

The menu item 'Select subset by range' is connected to the function selectSubset. The search pattern is treated as a selection of sample sequence numbers (effectively a list separated by commas that may also contain ranges expressed by colons). The current data will be replaced by its newly chosen subset.

Value

Overwrites the data frame 'labels' and numeric matrix 'WR' by subset that fulfills the search criteria.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
## Not run:
Search pattern = 1:5
# First to fifth samples in the data set

Search pattern = 1,10
# First and tenth samples

Search pattern = 1:5, 10:11, 25
# Samples number 1, 2, ...5, 10, 11, 25

## End(Not run)
```

---

summaryAll *Statistics: Statistical summaries for the whole data set or its subset*

Description

The function 'summaryAll' prints statistical summary for selected list of elements (majors as a default) and the current dataset (or its part). Function 'summaryMajor' is an entry point supplying the default list for major elements.

Usage

```
summaryAll(elems = major, where = NULL, show.boxplot = FALSE,
    show.hist = FALSE, silent=TRUE)
summaryMajor()
```

Arguments

elems	list of desired elements
where	list of desired samples to be processed
show.boxplot	logical, should be plotted the boxplots?
show.hist	logical, should be plotted the histograms?
silent	logical, should be the above chosen by the appropriate dialogues?

Details

The statistical summary involves number of observations, missing values, mean, standard deviation, minimum, 25% quartile, median (= 50% quartile), 75% quartile and maximum. The function also plots summary boxplots and histograms, if desired so.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSamples for details.

Even though as a default are assumed majors (SiO2,TiO2,Al2O3,FeOt,MnO,MgO,CaO,Na2O,K2O for 'summaryMajor'), the variable(s) to be displayed can be modified/specified. To this purpose serves the function 'selectColumnsLabels'.

In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

`results` numeric matrix with the results

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

statistics

summarySingle

summarySingleByGroup

summaryByGroup

Examples

```
data(combined)
accessVar("combined")

summaryAll(major)

summaryAll(major,show.hist=TRUE)

summaryAll(major,show.boxplot=TRUE)

# user-defined list
my.elems<-c("SiO2","K2O/Na2O","MgO")
summaryAll(my.elems)

## Not run:
    summaryMajor()

## End(Not run)
```

`summaryByGroup` *Statistics: Statistical summaries by groups*

Description

The function 'summaryByGroup' prints a statistical summary for selected list of elements (majors as a default) and the whole dataset or its selection, respecting the current grouping. Functions 'summaryByGroupMjr' and 'summaryByGroupTrc' are entry points supplying the default lists for major- and trace elements. The function 'summaryByGroupTrc' returns only ranges of the given parameter(s).

Usage

```
summaryByGroup(elems = major, where = NULL, show.boxplot = FALSE,
    show.hist = FALSE, silent = TRUE)

summaryByGroupMjr()
```

```
summaryByGroupTrc()

summaryRangesByGroup(elems=major, where=NULL, silent=TRUE)
```

Arguments

elems	list of desired elements
where	list of desired samples to be processed
show.boxplot	logical, should be plotted the boxplots?
show.hist	logical, should be plotted the histograms?
silent	logical, should be the above chosen by the appropriate dialogues?

Details

The statistical summary involves number of observations, missing values, mean, standard deviation, minimum, 25% quartile, median (= 50% quartile), 75% quartile and maximum. The function also plots a summary boxplots and histograms, if desired so.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSamples for details.

The defaults are lists of major (SiO2,TiO2,Al2O3,FeOt,MnO,MgO,CaO,Na2O, K2O) or trace (Rb,Sr,Ba,Cr,Ni,La,Eu,Y,Zr) elements, respectively.

The desired variables are selected using the function 'selectColumnsLabels'.

In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

results	a list with the results for individual groups

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

Examples

```
data(blatna)
accessVar("blatna")
groupsByLabel("Suite")

summaryByGroup(LILE)

summaryByGroup(LILE,show.hist=TRUE)

summaryByGroup(LILE,show.boxplot=TRUE)

# user-defined list
my.elems<-c("Rb","Sr","Ba/Sr")
summaryByGroup(my.elems)

summaryRangesByGroup(elems="Rb/Sr,Na2O+K2O")
```

```
    ## Not run:
        summaryByGroupTrc()
        summaryByGroupMjr()

## End(Not run)
```

---

summarySingle *Statistics: Single variable all/selection*

Description

Prints statistical summary for a single variable and the current dataset (or its part).

Usage

```
summarySingle(xlab="")
```

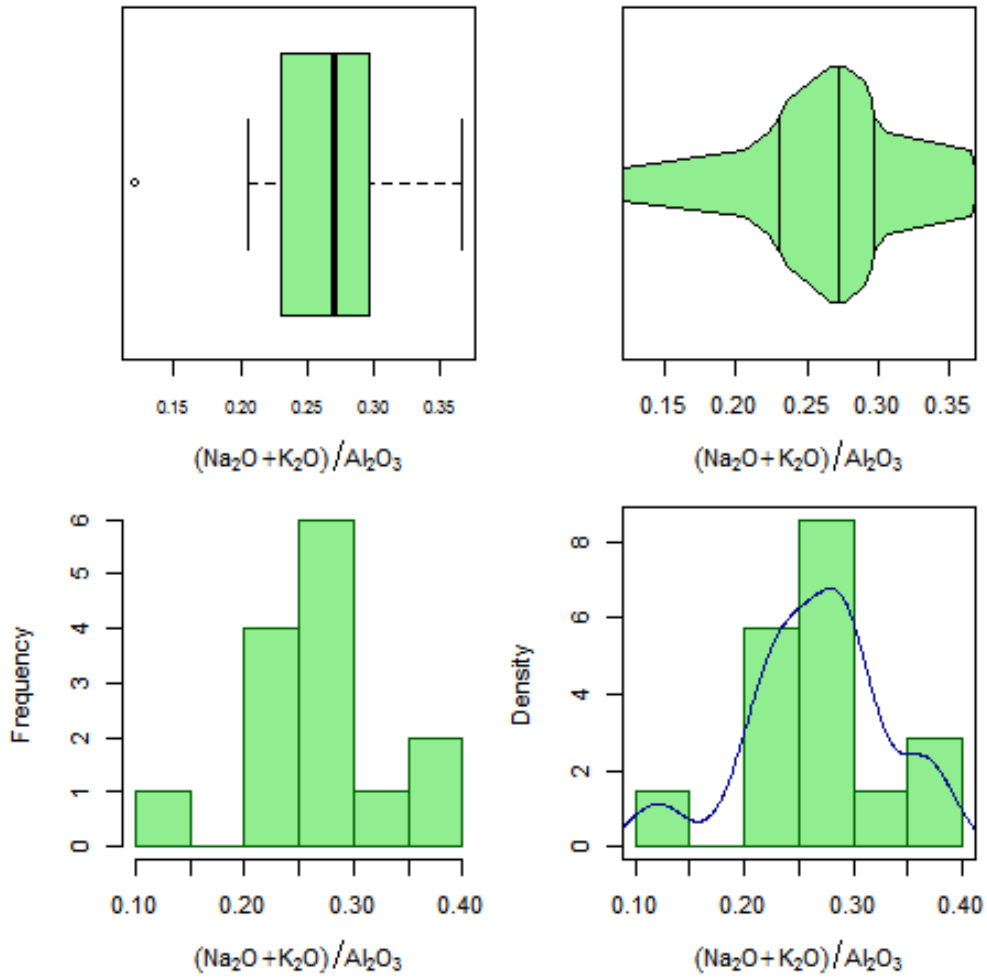
Arguments

xlab	variable name

Details

The statistical summary involves number of observations, missing values, mean, standard deviation, minimum, 25% quartile, median (=50% quartile), 75% quartile and maximum. The function also plots a summary boxplot and histogram.

In addition the statistical distribution of the given variable is shown as a boxplot, a box-percentile plot and two variants of histograms.



If no variable is specified as an argument 'xlab', the user can enter it using the function 'selectColumnLabel'. In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

The samples can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSamples for details.

Value

`results` numeric matrix/vector with the results

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

boxplot

bpplot2

statistics

summarySingleByGroup summaryAll summaryByGroup

Examples

```
data(blatna)
accessVar("blatna")

summarySingle("(Na2O+K2O)/Al2O3")
```

---

summarySingleByGroup *Statistics: Single variable by groups*

Description

Prints statistical summary for a single variable and the whole dataset, divided by groups.

Usage

```
summarySingleByGroup(xlab="")
```

Arguments

xlab	variable name

Details

The statistical summary involves number of observations, missing values, mean, standard deviation, minimum, 25% quartile, median (= 50% quartile), 75% quartile and maximum. The function also plots a summary boxplot and histogram.

If no variable is specified as an argument 'xlab', the user can enter it using the function 'selectColumnLabel'. In the specification of the variable can be used also arithmetic expressions, see calcCore for the correct syntax.

Value

results	numeric matrix with the results

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

boxplot summarySingle statistics summaryAll summaryByGroup

Examples

```
data(sazava)
accessVar("sazava")
groupsByLabel("Intrusion")

summarySingleByGroup("(Na2O+K2O)/Al2O3")
```

ternary *Ternary plot*

Description

These functions plot/add data to a ternary plot.

Usage

```
ternary(x = NULL, y = NULL, z = NULL, samples = rownames(WR),
    new = TRUE, grid = FALSE, ticks = TRUE, ...)

triplot(aa, bb, cc, alab, blab, clab, title = "", grid.int = 0,
tick.int = 0, label.axes = FALSE, line = FALSE,
pch = labels[names(aa), "Symbol"],
col = labels[names(aa), "Colour"],
cex = labels[names(aa),"Size"],
identify = getOption("gcd.ident"),
new = TRUE,...)

triplotadd(aa, bb, cc,
pch=labels[names(aa),"Symbol"],
col=labels[names(aa),"Colour"],
cex = labels[names(aa),"Size"],
labs=NULL, identify = FALSE, lines = FALSE, lty = "solid", type="p", lwd = 1)
```

Arguments

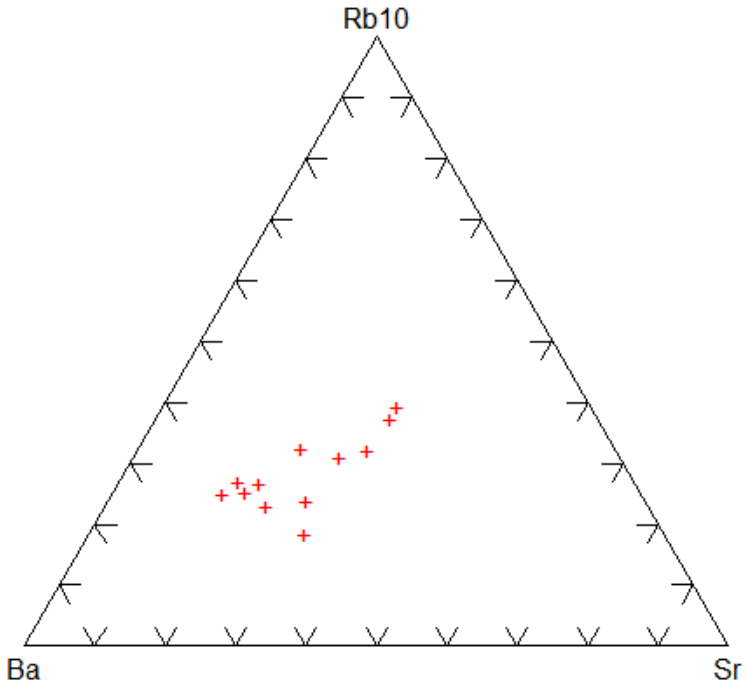
x	character; specification of the plotting variable for the bottom left apex (formulae OK).
y	character; specification of the plotting variable for the top apex (formulae OK).
z	character; specification of the plotting variables for the bottom right apex (formulae OK).
grid	logical; should be grid plotted?
ticks	logical; should be ticks plotted?
samples	character or numeric vector; specification of the samples to be plotted.
new	logical; should be opened a new plotting window?
...	Further parameters to the functions 'ternary' and 'triplot'.
aa	a numerical vector, bottom left apex.
bb	a numerical vector, top apex.
cc	a numerical vector, bottom right apex.
alab,blab,clab	labels for the apices.
title	title for the whole diagram.
grid.int	interval of grid lines (0-1); if set to zero (default value), no grid is drawn.
tick.int	interval of ticks on axes (0-1); if set to zero (default value), no ticks are drawn.
label.axes	logical; if set to TRUE, axes are labeled by percentages of the components.

`line, lines`	logical; if set to TRUE, lines are drawn instead of plotting points.
`lty`	line type.
`lwd`	line width.
`pch`	plotting symbols.
`col`	plotting colours.
`cex`	relative size of plotting symbols.
`identify`	logical; should be samples identified?
`labs`	character; optional text to label the points.
`type`	character; plot type; see plot.default.

Details

The function '`ternary`' is the user interface to '`triplot`'. The latter sets up the axes, labels the apices, plots the data and, if desired, enables the user to identify the data points interactively.

If '`new=TRUE`', new plot window is opened.



The values for '`label.axes`' are chosen according to '`tick.int`' or '`grid.int`'; if these are not available, labels are drawn by 10%.

'`triplotadd` 'adds data points/lines to pre-existing ternary plot.

The variables to be plotted are selected using the function '`selectColumnLabel`.

In the specification of the apices can be used also arithmetic expressions, see `calcCore` for the correct syntax.

The functions are Figaro-compatible.

Value

A numeric matrix with coordinates of the data points recast to a sum of 1.

Author(s)

Jakub Šmíd <smid@prfdec.natur.cuni.cz> & Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

plot

Examples

```
data(sazava)
accessVar("sazava")

ternary("Ba","Rb*10","Sr",col="red",pch="+")

ternary("SiO2/10","2*FeOt","K2O*5",samples=1:10,grid=TRUE)

triplot(WR[,"SiO2"]/10,WR[,"Na2O"]+WR[,"K2O"],WR[,"MgO"],"SiO2","A","MgO",
    tick.int=0.1)

triplot(WR[,"Rb"]*10,WR[,"Sr"],WR[,"Ba"],"Rb","Sr","Ba",tick.int=0.05,
    grid.int=0.1,pch="+",col="darkblue",label.axes=TRUE)
```

---

threeD *3D plot*

Description

Plots a 3-D plot of three specified variables.

Usage

```
threeD(xlab="",ylab="",zlab="")
```

Arguments

xlab	Name of the data column to be used as x axis.
ylab	Name of the data column to be used as y axis.
zlab	Name of the data column to be used as z axis.

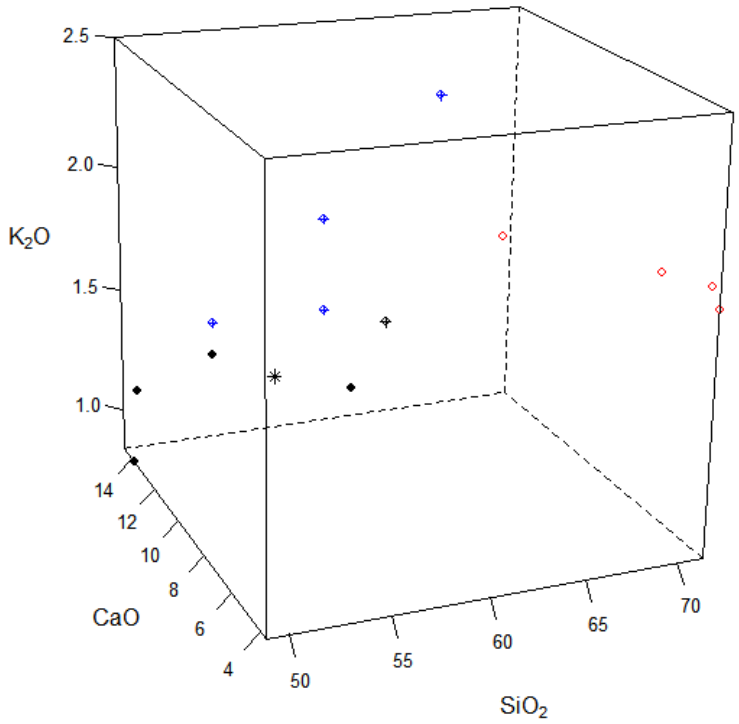
Details

This function displays three variables in a form of 3D plot. The plot can be rotated interactively, if required so.

The samples to be plotted can be selected based on combination of three searching mechanisms (by sample name/label, range or a Boolean condition) - see selectSubset for details.

If no parameters 'xlab','ylab' and 'zlab' are given, the user is prompted to specify them.

The variables are selected using the function 'selectColumnLabel.



In the specification of the apices can be used also arithmetic expressions, see `calcCore` for the correct syntax.

See manual entry for '`cloud`' for further details.

Value

None.

Warning

This function IS NOT Figaro-compatible.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz> & Vojtěch Erban, <erban@sopky.cz>

Examples

```
data(atacazo)
accessVar("atacazo")

threeD("SiO2","Na2O+K2O","MgO+FeOt")
```

tkSelectVariable *TclTk GUI: Select a single variable*

Description

Function to select a single variable using the Tcl/Tk-based Graphical User Interface (GUI).

Usage

```
tkSelectVariable(top.frame = NULL, where = colnames(WR), preselect = 2,
    pack = FALSE, message = "Select a variable", background = "wheat",
    variable = "x", on.leave = function() {}, row = 0, column = 0, height = 15,
    width = 50, buttons = FALSE, state = "normal")
```

Arguments

top.frame	name of the parental frame
where	character; names of variables to be chosen from
preselect	numeric; which item is to be preselected
pack	logical; pack the frame?
message	character; textual prompt
background	colour for the frame background
variable	character; variable name with the output
on.leave	function to be invoked upon leave
row, column	coordinates within the parental frame
height, width	size of the frame
buttons	logical; should the frame have also buttons?
state	character; either 'normal' or 'disabled'

Details

The buttons are: Reset, SortUp, SortDown, OK, Cancel.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

tcltk-package

tk_winDialog *tk_winDialog*

Description

Tcl/Tk replacement for the MS Windows-specific function 'winDialog'.

Usage

```
tk_winDialog(type="ok",message="")
```

Arguments

type	Character; the type of the dialogue box.
message	Character. The information field of the dialogue box.

Details

This is a platform-independent implementation of the MS Windows-specific function 'winDialog', written using the Tcl/Tk. Possible types of the dialogue box are: ok, okcancel, yesno and yesnocancel.

Value

A character string giving the name of the button pressed (in capitals).

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

winDialog tkmessageBox tk_winDialogString tcltk-package

Examples

```
## Not run:
tk_winDialog(type="yesnocancel",message="Are you sure?")

## End(Not run)
```

tk_winDialogString *tk_winDialogString*

Description

Tcl/Tk replacement for the MS Windows-specific function 'winDialogString'.

Usage

```
tk_winDialogString(message="Enter variable",default="",returnValOnCancel=NULL)
```

Arguments

message	Character. The information field of the dialog box.
default	Character; the default string.
returnValOnCancel	Character; a value to be returned when the dialogue is canceled.

Details

This is a platform-independent implementation of the MS Windows-specific function 'winDialogString', written using the Tcl/Tk.

Value

A character string giving the contents of the text box when Ok was pressed, or value specified by 'returnValOnCancel' if Cancel was pressed.

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

See Also

winDialogString tkentry tk_winDialog tcltk-package

Examples

```
## Not run:
tk_winDialogString(message="Enter x value",default="15.7")

## End(Not run)
```

`trendTicks` *Petrogenetic trends*

Description

Adding a trend with arrow and tick marks to a pre-existing GCDkit plot.

Usage

```
trendTicks(equation=NULL, x, y = NULL,
    xmin = par("usr")[1], xmax = par("usr")[2],
    tick = abs(par("tcl")), col = "blue", lty = "solid",
    lwd = 1, arrow = FALSE, text = "", text.adj = c(1,0.5),
    plot=TRUE, autoscale = TRUE)
```

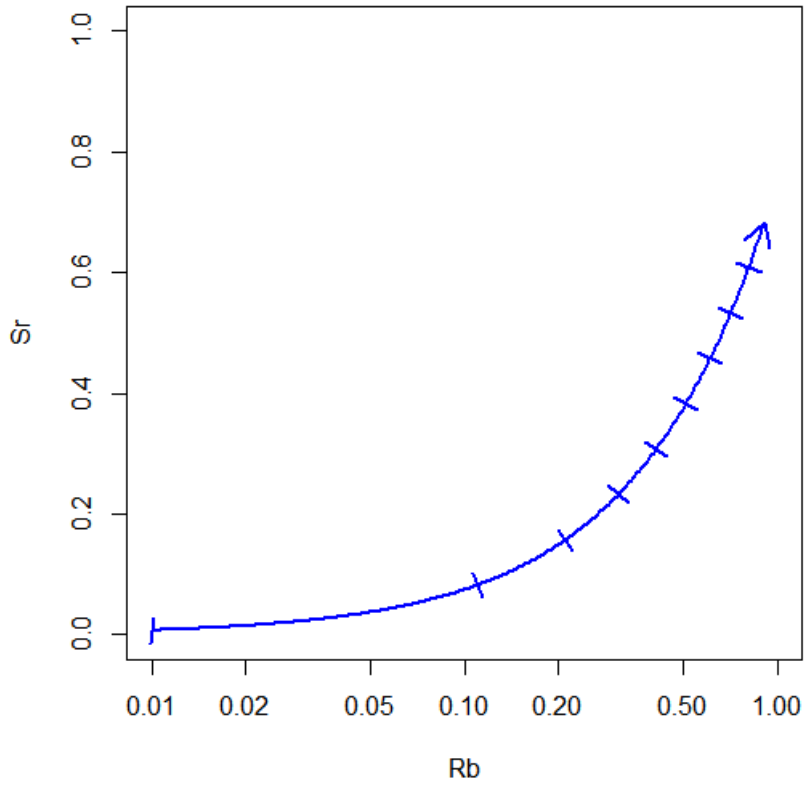
Arguments

`equation`	character or expression; a valid formula expressed as a function of `x`.
`x`	numeric; x values where the ticks are to be drawn.
`y`	numeric; (optional) y values where the ticks are to be drawn.
`xmin`	numeric; beginning of the trend.
`xmax`	numeric; end of the trend.
`tick`	numeric; length of a tick as a fraction of the height of a line of text.
`col`	text or numeric; plotting colour specification.
`lty`	text or numeric; the line type.
`lwd`	numeric; the line width, a positive number, defaulting to 1.
`arrow`	logical; should be also an arrow head shown?
`text`	character; (optional) labels for individual ticks.
`text.adj`	adjustment of the text. See `par`.
`plot`	logical; should be the trend plotted?
`autoscale`	logical; should the plot be autosized in order to accommodate the whole trend as well as all data points?

Details

Using the function `curve`, the function `trendTicks` adds to an existing GCDkit plot a linear or curved trend with tick marks and (optionally) arrow head. If `equation` is provided, it is required that the trend is defined as a function of `x`. Otherwise, a fourth-order polynomial is fitted to the `[x,y]` data. If `plot = FALSE`, no trend is plotted (and only the calculations are performed).

The slope of the individual tick marks is determined using a numerical derivative of the main function at the respective points.



Value

Returns (invisibly) a list with the following components:

`equation`	expression, specified/fitted equation for the trend,
`results`	coordinates of the points from which the ticks are drawn, if `[x,y]` (and not formula of the trend) was specified, there is also a result of fourth-order polynomial fit (`y_estd.`),
`x, y`	`[x,y]` coordinates of the points from which the ticks are drawn,
`slopes`	slopes of the tick lines,
`ticks`	numeric matrix; `[x1,y1]` and `[x2,y2]` coordinates of the end points of individual ticks,
`text.labels`	textual labels to individual ticks; list of parameters to the function `text`,
`arrow.head`	numeric matrix, [x,y] coordinates of the arrow head.

Warning

Autoscaling will work only with Figaro compatible plots!

Author(s)

Vojtěch Janoušek, `<vojtech.janousek@geology.cz>`

See Also

par lm D

Examples

```
data(sazava)
accessVar("sazava")

# EXAMPLE 1
# Equation provided, real data, no autoscaling
binary("Ba","Sr",xmin=200,xmax=2000,ymin=10,ymax=650)
figCex(1.5)
equation<-"x/8+200"
x<-seq(2000,500,by=-100)
out<-trendTicks(equation,x=x,xmin=min(x),xmax=max(x),col="darkred",lty="solid",
    lwd=2,arrow=TRUE,autoscale=TRUE)

# EXAMPLE 2
# Just the trend, autoscaled, x axis is logarithmic
windows()
plot(1,1,type="n",xlim=c(50,150),ylim=c(50,250),xlab="Rb",ylab="Sr",log="xy")
equation<-"15*x/8+10"
x<-seq(50,120,length=10)
trendTicks(equation,x=x,xmin=min(x),xmax=max(x),col=2,lwd=2,
    arrow=FALSE,autoscale=FALSE)

# EXAMPLE 3
# Calculate Rayleigh-type fractionation trend
ff<-seq(1,0.1,-0.1) # F, amount of melt left
x<-80*ff^(1.2-1)    # cL for three elements, arbitrary D of 1.2, 2.0 and 1.3
y<-550*ff^(2.0-1)
z<-1000*ff^(1.3-1)
my.trend<-cbind(x,y,z)
colnames(my.trend)<-c("Rb","Sr","Ba")
rownames(my.trend)<-ff

# No equation provided, just [x,y] data are given
# Linear coordinates, autoscaled to accommodate both data and trend
binary("Rb","Sr",log="",xaxs="r",yaxs="r")
out<-trendTicks(equation=NULL,x=x,y=y,xmin=min(x),xmax=max(x),col="red",text=ff)

# Linear coordinates, not autoscaled
binary("Rb","Sr",log="",xaxs="r",yaxs="r")
out<-trendTicks(equation=NULL,x=x,y=y,xmin=min(x),xmax=max(x),col="red",text=ff,
    autoscale=FALSE)

# The same, no trend plotting (just calculating for later use)
binary("Rb","Sr",log="",xaxs="r",yaxs="r",xmin=20,xmax=100,ymin=10,ymax=700)
out<-trendTicks(equation=NULL,x=x,y=y,xmin=20,xmax=100,col="red",text=ff,
    arrow=TRUE,plot=FALSE)

# Manual overplotting of the trend from the object 'out'
# Points
points(out$results["x",],out$results["y_obs.",],col="red",pch="+",cex=2)

# Trend curve
```

```
figRedraw()
.curveMy(out$equation,from=min(out$results["x",],na.rm=TRUE),to=max(out$results["x",],na.rm=TRUE),
    col="red",lty="solid",lwd=2)

# Tick marks
segments(out$ticks[,"x1"],out$ticks[,"y1"],out$ticks[,"x2"],out$ticks[,"y2"],col="red",lwd=2)

# Arrow head
lines(out$arrow.head,col="red",lwd=2)

# Textual labels, no rotation
text(out$text.labels$x,out$text.labels$y,out$text.labels$text,pos=3)

# EXAMPLE 4
# Logarithmic coordinates
binary("Rb","Sr",log="x")
trendTicks(equation=NULL,x=x,y=y,xmin=min(x),xmax=max(x),col="red",text=ff,autoscale=TRUE)
```

`webMineral` *Search minerals by chemistry (webmineral.com)*

Description

Front-end to the webmineral.com database.

Usage

```
webMineral()
```

Arguments

None.

Details

This function opens a web browser and simply connects to the database of webmineral.com. The Search form allows: (1) to display the listing of minerals containing an element selected from the periodic chart, sorted by wt.% (2) to search Webmineral's chemical compositional table using the Composition Search Form.

Value

None.

Author(s)

The database is by David Barthelmy, <dbarthelmy@webmineral.com>

`wholeRock` *Liquid components of* Putirka (2008)

Description

This function recalculates whole-rock analyses into liquid components after *Putirka (2008)* (see his Tab. 1).

Usage

```
wholeRock()
```

Details

This auxiliary function returns liquid components, i.e. parameters needed for thermobarometric calculations presented by *Putirka (2008)*. The calculated values include:

(1) Mole proportions of oxides
(2) Mole fractions
(3) Cation proportions
(4) Cation fractions
(5) *Beattie (1993)* components (CSiO2, CNM, CNF, NF)

The following formulae are implemented:

Mole proportions:

$$\frac{Oxide_i(wt.\%)}{MW(Oxide_i)}$$

Mole fractions: ditto, but recast to a sum of 1

Cation proportions:

$$\frac{Oxide_i(wt.\%)}{MW(Oxide_i)} \times no.cats(Oxide_i)$$

Cation fractions: ditto, but recast to a sum of 1

Beattie (1993) components are based on cation fractions and defined as follows:

$$C^{liq}_{SiO_2} = X^{liq}_{SiO_2}$$

$$C^{liq}_{NM} = X^{liq}_{FeO} + X^{liq}_{MnO} + X^{liq}_{MgO} + X^{liq}_{CaO} + X^{liq}_{CoO} + X^{liq}_{NiO}$$

$$C^{liq}_{NF} = X^{liq}_{SiO_2} + X^{liq}_{NaO_{0.5}} + X^{liq}_{KO_{0.5}} + X^{liq}_{TiO_2}$$

$$NF = \frac{7}{2}\ln(1 - X^{liq}_{AlO_{1.5}}) + 7\ln(1 - X^{liq}_{TiO_2})$$

Value

The function assigns a global variable `WR.recalc`, i.e. a list with components (see also Details):

`mole.prop`	Mole proportions
`mole.fract`	Mole fractions
`cat.prop`	Cation proportions
`cat.fract`	Cation fractions
`beattie`	*Beattie (1993)* components

Author(s)

Vojtěch Janoušek, <vojtech.janousek@geology.cz>

References

Beattie P (1993) Olivine-melt and orthopyroxene-melt equilibria. Contrib Mineral Petrol 115:103-111 doi: 10.1007/BF00712982doi: 10.1007/BF00712982

Putirka KD (2008) Thermometers and barometers for volcanic systems. In: Putirka KD, Tepley III FJ (eds) Minerals, Inclusions And Volcanic Processes. Mineralogical Society of America and Geochemical Society Reviews in Mineralogy and Geochemistry 69, Washington, pp 61-120 doi: 10.2138/rmg.2008.69.3doi: 10.2138/rmg.2008.69.3

Examples

```
# Analysis of Putirka (2008) - see his Tab. 1
WR<-matrix(c(47.05,0.89,16.11,7.96,0.15,12.78,11.13,2.18,0.04,0,0,0,0),nrow=1)
colnames(WR)<-c("SiO2","TiO2","Al2O3","FeO","MnO","MgO","CaO","Na2O",
    "K2O","Cr2O3","CoO","NiO","P2O5")
rownames(WR)<-"Putirka_test"

wholeRock()
print(WR.recalc,digits=4)
```

Index

∗ **Calculation**
calc, 25
calcCore, 26
clr.transform, 27
formula2vector, 65
minComp, 85
Molecular weights, 99
oxide2oxide, 104
oxide2ppm, 105
ppm2oxide, 123
printSamples, 125
printSingle, 127
r2clipboard, 133
recast, 135
saveResults, 139
statsByGroup, 159
summaryAll, 165
summaryByGroup, 166
summarySingle, 168
summarySingleByGroup, 170
∗ **Classification**
cutMy, 37
figGbo, 54
∗ **Edit**
assign1col, 10
assign1symb, 11
assignColLab, 11
assignColVar, 13
assignSymbGroup, 14
assignSymbLab, 15
assignSymbLett, 16
Boolean conditions, 22
calc, 25
calcCore, 26
cutMy, 37
Edit labels, 38
Edit numeric data, 38
editLabFactor, 39
groupsByLabel, 70
joinGroups, 73
peterplot, 110
phasePropPlot, 112
recast, 135
Regular expressions, 136
selectAll, 143
selectByLabel, 144
selectColumnLabel, 146
selectColumnsLabels, 147
selectPalette, 149
selectSubset, 151
setTransparency, 155
Subset by range, 164
tk_winDialog, 176
tk_winDialogString, 177
tkSelectVariable, 175
∗ **Figaro**
Add contours, 7
contourAll, 32
contourGroups, 34
figAdd, 46
figAddReservoirs, 49
figaro.identify, 51
figCol, 52
figEdit, 53
figGbo, 54
figMulti, 54
figOverplot, 56
figRedraw, 58
figScale, 59
figUser, 60
figZoom, 61
filledContourFig, 63
∗ **File**
Export to Access, 40
Export to DBF, 41
Export to Excel, 42
Export to HTML tables, 43
loadData, 74
mergeData, 78
pdfAll, 108
peekDataset, 109
pokeDataset, 122
psAll, 130
r2clipboard, 133
saveData, 138
saveResults, 139

∗ **GUI**
tk_winDialog, 176
tk_winDialogString, 177
tkSelectVariable, 175
∗ **Grouping**
assignSymbGroup, 14
assignSymbLab, 15
assignSymbLett, 16
cutMy, 37
figGbo, 54
figMulti, 54
groupsByLabel, 70
joinGroups, 73
statsByGroup, 159
statsByGroupPlot, 160
strip, 161
stripBoxplot, 162
summaryByGroup, 166
summarySingleByGroup, 170
∗ **Menu: Calculations: Statistics**
cluster, 30
pairsCorr, 105
prComp, 124
statsByGroup, 159
statsByGroupPlot, 160
strip, 161
stripBoxplot, 162
summaryAll, 165
summaryByGroup, 166
summarySingle, 168
summarySingleByGroup, 170
∗ **Menu: Calculations**
calc, 25
Export to Access, 40
Export to DBF, 41
Export to Excel, 42
Export to HTML tables, 43
r2clipboard, 133
recast, 135
saveResults, 139
∗ **Menu: Data handling**
assign1col, 10
assign1symb, 11
assignColLab, 11
assignColVar, 13
assignSymbGroup, 14
assignSymbLab, 15
assignSymbLett, 16
Boolean conditions, 22
cutMy, 37
Edit labels, 38
Edit numeric data, 38
editLabFactor, 39
Export to Access, 40
Export to DBF, 41
Export to Excel, 42
Export to HTML tables, 43
groupsByLabel, 70
joinGroups, 73
phasePropPlot, 112
printSamples, 125
printSingle, 127
selectAll, 143
selectByLabel, 144
selectColumnLabel, 146
selectColumnsLabels, 147
selectPalette, 149
selectSubset, 151
setTransparency, 155
showColours, 156
showLegend, 157
showSymbols, 159
Subset by range, 164
∗ **Menu: GCDkit**
about, 5
accessVar, 6
crosstab, 36
gcdOptions, 66
graphicsOff, 69
info, 73
loadData, 74
mergeData, 78
pdfAll, 108
peekDataset, 109
pokeDataset, 122
psAll, 130
purgeDatasets, 131
saveData, 138
setCex, 153
setShutUp, 154
∗ **Menu: Modelling**
trendTicks, 178
∗ **Menu: Plot apfu**
.callGCDkit, 4
∗ **Menu: Plot editing**
Add contours, 7
contourAll, 32
contourGroups, 34
figAdd, 46
figAddReservoirs, 49
figaro.identify, 51
figCol, 52
figEdit, 53
figGbo, 54

figMulti, 54
figOverplot, 56
figRedraw, 58
figScale, 59
figUser, 60
figZoom, 61
filledContourFig, 63
∗ **Menu: Plots**
binary, 18
binaryBoxplot, 20
bpplot2, 23
ID, 72
Multiple plots, 100
Plate, 114
Plate editing, 116
plateLabelSlots, 118
plotWithCircles, 120
profiler, 128
ternary, 171
threeD, 173
∗ **Menu: Plot**
peterplot, 110
∗ **Menu: Plugins**
clr.transform, 27
scattersmooth, 140
∗ **Modelling**
trendTicks, 178
∗ **Parameter**
gcdOptions, 66
setCex, 153
setShutUp, 154
∗ **Plates**
Plate, 114
Plate editing, 116
plateLabelSlots, 118
∗ **Plot**
.callGCDkit, 4
Add contours, 7
assign1col, 10
assign1symb, 11
assignColLab, 11
assignColVar, 13
assignSymbGroup, 14
assignSymbLab, 15
assignSymbLett, 16
binary, 18
binaryBoxplot, 20
bpplot2, 23
cluster, 30
contourAll, 32
contourGroups, 34
figAdd, 46
figAddReservoirs, 49
figaro.identify, 51
figCol, 52
figEdit, 53
figGbo, 54
figMulti, 54
figOverplot, 56
figRedraw, 58
figScale, 59
figUser, 60
figZoom, 61
filledContourFig, 63
gcdOptions, 66
graphicsOff, 69
ID, 72
Multiple plots, 100
pairsCorr, 105
pdfAll, 108
peterplot, 110
phasePropPlot, 112
Plate, 114
Plate editing, 116
plateLabelSlots, 118
plotWithCircles, 120
prComp, 124
profiler, 128
psAll, 130
scattersmooth, 140
selectPalette, 149
setCex, 153
setTransparency, 155
showColours, 156
showLegend, 157
showSymbols, 159
statsByGroupPlot, 160
strip, 161
stripBoxplot, 162
summaryAll, 165
summaryByGroup, 166
summarySingle, 168
summarySingleByGroup, 170
ternary, 171
threeD, 173
∗ **Plugin: disclosure**
clr.transform, 27
∗ **Print**
calc, 25
calcCore, 26
crosstab, 36
Export to Access, 40
Export to DBF, 41
Export to Excel, 42

Export to HTML tables, 43
info, 73
printSamples, 125
printSingle, 127
setShutUp, 154
∗ **Profile plotting**
profiler, 128
∗ **Spiderplot**
bpplot2, 23
∗ **Statistics**
cluster, 30
filledContourFig, 63
pairsCorr, 105
prComp, 124
statsByGroup, 159
statsByGroupPlot, 160
strip, 161
stripBoxplot, 162
summaryAll, 165
summaryByGroup, 166
summarySingle, 168
summarySingleByGroup, 170
∗ **Subsetting**
Boolean conditions, 22
selectAll, 143
selectByLabel, 144
selectColumnLabel, 146
selectColumnsLabels, 147
selectSubset, 151
Subset by range, 164
tkSelectVariable, 175
∗ **Switching datasets**
peekDataset, 109
pokeDataset, 122
purgeDatasets, 131
∗ **TclTk**
tk_winDialog, 176
tk_winDialogString, 177
tkSelectVariable, 175
∗ **aplot**
Add contours, 7
assign1symb, 11
assignSymbGroup, 14
assignSymbLab, 15
assignSymbLett, 16
contourAll, 32
contourGroups, 34
figAdd, 46
figAddReservoirs, 49
figCol, 52
figEdit, 53
figOverplot, 56
figScale, 59
figUser, 60
figZoom, 61
peterplot, 110
showColours, 156
showSymbols, 159
∗ **classes**
HTMLFormula, 70
minAllocateAtoms, 79
minAssign, 81
minCheckValency, 83
minClassify, 84
minEndMembers, 87
mineral-class, 89
mineral.db-class, 92
minFormula, 93
minValues, 98
recalcOptions, 134
selectByMineral, 145
∗ **closure**
clr.transform, 27
∗ **cluster**
cluster, 30
∗ **color**
assign1col, 10
assignColLab, 11
assignColVar, 13
figAdd, 46
figAddReservoirs, 49
figCol, 52
figOverplot, 56
figUser, 60
selectPalette, 149
setTransparency, 155
showColours, 156
∗ **compositional data**
clr.transform, 27
∗ **database**
Boolean conditions, 22
Regular expressions, 136
selectColumnLabel, 146
selectColumnsLabels, 147
selectSubset, 151
Subset by range, 164
∗ **datasets**
alumosilicates, 8
amphibole, 9
apatite, 9
atacazo, 17
blatna, 21
combined, 31
feldspars, 46

garnet, 66
micas, 79
olivine, 103
pyroxene, 132
sazava, 139
∗ **device**
figRedraw, 58
pdfAll, 108
psAll, 130
∗ **dplot**
setCex, 153
∗ **file**
Export to Access, 40
Export to DBF, 41
Export to Excel, 42
Export to HTML tables, 43
loadData, 74
mergeData, 78
pdfAll, 108
peekDataset, 109
pokeDataset, 122
psAll, 130
purgeDatasets, 131
r2clipboard, 133
saveData, 138
saveResults, 139
∗ **hplot**
.callGCDkit, 4
binary, 18
binaryBoxplot, 20
bpplot2, 23
cluster, 30
figMulti, 54
filledContourFig, 63
graphicsOff, 69
Multiple plots, 100
pairsCorr, 105
phasePropPlot, 112
Plate, 114
Plate editing, 116
plateLabelSlots, 118
plotWithCircles, 120
profiler, 128
scattersmooth, 140
statsByGroupPlot, 160
strip, 161
stripBoxplot, 162
summaryAll, 165
summaryByGroup, 166
summarySingle, 168
summarySingleByGroup, 170
ternary, 171
threeD, 173
∗ **iplot**
contourAll, 32
contourGroups, 34
figAdd, 46
figAddReservoirs, 49
figaro.identify, 51
figCol, 52
figEdit, 53
figGbo, 54
figOverplot, 56
figUser, 60
gcdOptions, 66
ID, 72
showLegend, 157
∗ **manip**
accessVar, 6
Boolean conditions, 22
calc, 25
calcCore, 26
clr.transform, 27
cutMy, 37
Edit labels, 38
Edit numeric data, 38
editLabFactor, 39
formula2vector, 65
groupsByLabel, 70
HTMLFormula, 70
joinGroups, 73
minAllocateAtoms, 79
minAssign, 81
minCheckValency, 83
minClassify, 84
minComp, 85
minDat, 87
minEndMembers, 87
minFormula, 93
minMain, 96
minValues, 98
Molecular weights, 99
oxide2oxide, 104
oxide2ppm, 105
ppm2oxide, 123
recast, 135
selectAll, 143
selectByLabel, 144
selectColumnLabel, 146
selectColumnsLabels, 147
selectSubset, 151
Subset by range, 164
tk_winDialog, 176
tk_winDialogString, 177

tkSelectVariable, 175
trendTicks, 178
webMineral, 181
wholeRock, 182
∗ **misc**
about, 5
crosstab, 36
info, 73
quitGCDkit, 133
∗ **multivariate**
cluster, 30
pairsCorr, 105
prComp, 124
∗ **print**
printSamples, 125
printSingle, 127
setShutUp, 154
∗ **univar**
statsByGroup, 159
statsByGroupPlot, 160
strip, 161
stripBoxplot, 162
summaryAll, 165
summaryByGroup, 166
summarySingle, 168
summarySingleByGroup, 170
∗ **utils**
minDat, 87
webMineral, 181
.callGCDkit, 4

about, 5
accessExport, *41, 42*
accessExport *(*Export to Access*)*, 40
accessVar, 6, *8–10*, *22*, *31*, *46*, *66*, *79*, *103*, *122, 123*, *131, 132*, *138*
Add contours, 7
addContours, *64*
addContours *(*Add contours*)*, 7
allFeII *(*minFormula*)*, 93
allFeII,mineral-method *(*mineral-class*)*, 89
allFeIII *(*minFormula*)*, 93
allFeIII,mineral-method *(*mineral-class*)*, 89
alumosilicates, 8
amphibole, 9
annotate, *53*
apatite, 9
assign1col, 10, *11, 12*, *14–16*, *155*, *158*
assign1symb, *10*, 11, *12*, *15, 16*, *158*
assignColLab, *10, 11*, 11, *14–16*, *150*, *155*, *158*
assignColVar, 13, *150*, *158*
assignSymbGroup, *10–12*, *14*, 14, *15, 16*, *155*, *158*
assignSymbLab, *10–12*, *15*, 15, *16*, *158*
assignSymbLett, *15*, 16
atacazo, 17

binary, *5*, 18, *55*, *115*
binaryBoxplot, *5*, 20
biplot.princomp, *125*
blatna, 21
blatna_iso *(*blatna*)*, 21
Boolean conditions, 22
boxplot, *21*, *24*, *164*, *169, 170*
bpplot2, 23, *169*

calc, 25
calcCore, *19, 20*, *25*, 26, *102*, *106*, *121*, *126, 127*, *129*, *142*, *161*, *163*, *165*, *167*, *169, 170*, *172*, *174*
chull, *33*, *35*
chullAll, *35*
chullAll *(*contourAll*)*, 32
chullGroups, *33*, *48*, *50*
chullGroups *(*contourGroups*)*, 34
cloud, *174*
clr.trans, *125*
clr.trans *(*clr.transform*)*, 27
clr.transform, 27
cluster, 30
colours, *48*, *50*, *52*, *138*, *157*
Colours: Plotting symbols *(*figCol*)*, 52
Colours: set to B&W *(*figCol*)*, 52
Colours: subtitle *(*figCol*)*, 52
Colours: title *(*figCol*)*, 52
combined, 31
contour, *7*, *32–35*
contourAll, 32, *35*
contourGroups, *33*, 34, *48*, *50*
cor, *107, 108*
cov, *107, 108*
crosstab, 36
curve, *47, 48*, *50*, *178*
cut, *37*
cutMy, 37

D, *180*
data, *8–10*, *22*, *31*, *46*, *66*, *79*, *103*, *132*
data.entry, *38, 39*
dbfExport, *40*, *42*
dbfExport *(*Export to DBF*)*, 41
dev.off, *69*
Droop *(*minFormula*)*, 93

Droop,mineral-method *(mineral-class)*, 89

Edit labels, 38
Edit numeric data, 38
Edit: subtitle *(figEdit)*, 53
Edit: title *(figEdit)*, 53
Edit: x label *(figEdit)*, 53
Edit: y label *(figEdit)*, 53
editData *(Edit numeric data)*, 38
editLabels *(Edit labels)*, 38
editLabFactor, 39
excel2007Export, *40*, *41*
excel2007Export *(Export to Excel)*, 42
excelExport, *40*, *41*
excelExport *(Export to Excel)*, 42
Export to Access, 40
Export to DBF, 41
Export to Excel, 42
Export to HTML tables, 43

FeAmph *(minFormula)*, 93
FeAmph,mineral-method *(mineral-class)*, 89
feldspars, 46
figAdd, 46
figAddArrow *(figAdd)*, 46
figAddBox *(figAdd)*, 46
figAddCurve *(figAdd)*, 46
figAddFit *(figAdd)*, 46
figAddReservoirs, *48*, 49, *57*
figAddText *(figAdd)*, 46
figAlab *(figEdit)*, 53
figaro, *5*, *7*, *33*, *35*, *48*, *50–55*, *57*, *59–62*, *64*, *69*, *85*, *102*, *103*, *113*, *115*, *117*, *119*
figaro.identify, 51
figBlab *(figEdit)*, 53
figBw *(figCol)*, 52
figCex *(figScale)*, 59
figCexLab *(figScale)*, 59
figCexMain *(figScale)*, 59
figCexSub *(figScale)*, 59
figClab *(figEdit)*, 53
figCol, 52, *117*
figColMain *(figCol)*, 52
figColours *(figCol)*, 52
figColSub *(figCol)*, 52
figEdit, 53
figFixLim *(figZoom)*, 61
figGbo, 54
figGrid *(figAdd)*, 46
figIdentify *(figaro.identify)*, 51
figLegend, *158*
figLegend *(figAdd)*, 46
figMain *(figEdit)*, 53
figMulti, 54, *115*
figOverplot, *47*, 56
figRedraw, 58
figScale, 59, *117*
figSub *(figEdit)*, 53
figTicks *(figAdd)*, 46
figUnzoom *(figZoom)*, 61
figUser, 60, *117*
figXlab *(figEdit)*, 53
figXlim *(figZoom)*, 61
figYlab *(figEdit)*, 53
figYlim *(figZoom)*, 61
figZoom, 61
figZooming *(figZoom)*, 61
filled.contour, *7*, *63*
filledContourFig, 63, *150*
FixedCats *(minFormula)*, 93
FixedCats,mineral-method *(mineral-class)*, 89
format,mineral.db-method *(mineral.db-class)*, 92
formula2vector, 65, *86*
formulaFixedAtoms *(minFormula)*, 93
formulaFixedAtoms,mineral-method *(mineral-class)*, 89
formulaFixedAtomsC *(minFormula)*, 93
formulaFixedCharges *(minFormula)*, 93
formulaFixedCharges,mineral-method *(mineral-class)*, 89
formulaFixedChargesC *(minFormula)*, 93
formulaFixedOxygens *(minFormula)*, 93
formulaFixedOxygens,mineral-method *(mineral-class)*, 89
formulaFixedOxygensC *(minFormula)*, 93

garnet, 66
gcdOptions, 66, *73*, *154*
getwd, *77*
graphicsOff, 69
groupsByLabel, 70

hclust, *31*
Highlight multiple points *(figaro.identify)*, 51
highlightSelection *(figaro.identify)*, 51
hist, *107*, *108*
HTMLFormula, 70
HTMLFormulaC *(HTMLFormula)*, 70
HTMLTableMain *(Export to HTML tables)*, 43
HTMLtableOrdered *(Export to HTML tables)*, 43

HTMLTableResults (Export to HTML tables), 43
HTMLTableWR (Export to HTML tables), 43

ID, *69*, 72
idealMineralCompositions, *65*, *86*
idealMineralCompositions (minComp), 85
identify, *51*, *69*, *73*
Identify points (figaro.identify), 51
info, 73

joinGroups, 73

kde2d, *7*, *33*, *35*

lda, *29*
lda.clr (clr.transform), 27
legend, *47*, *48*, *50*, *158*
lines, *128*
lm, *180*
loadData, 74, *78*, *122*, *123*, *131*, *138*, *144*
loadDataOdbc (loadData), 74
lowess, *107*, *108*

mean, *27*
menuet, *67*, *69*
merge, *78*
mergeData, *77*, 78, *138*
mergeDataCols (mergeData), 78
mergeDataRows (mergeData), 78
micas, 79
minAllocateAtoms, *72*, 79, *82*, *84*, *85*, *88–91*, *95–99*
minAllocateAtoms,mineral-method (mineral-class), 89
minAllocateAtomsAll (minAllocateAtoms), 79
minAllocateAtomsC (minAllocateAtoms), 79
minAssign, *71*, *80*, 81, *84*, *85*, *89*, *91*, *95*, *97*, *99*
minAssignC (minAssign), 81
minCheckValency, *71*, *80*, *82*, 83, *89–91*, *95*, *97*, *99*
minCheckValency,mineral-method (mineral-class), 89
minCheckValencyAll (minCheckValency), 83
minCheckValencyC (minCheckValency), 83
minClassify, *81*, *82*, *84*, 84, *89*, *91*, *95*, *97*, *99*
minClassifyPlot (minClassify), 84
minComp, *65*, 85, *86*
minDat, 87
minEndMembers, *72*, *81*, *82*, *84*, *85*, 87, *90*, *91*, *95–97*, *99*
minEndMembers,mineral-method (mineral-class), 89
minEndMembersAll (minEndMembers), 87
minEndMembersC (minEndMembers), 87
mineral-class, 89
mineral.db-class, 92
minFormula, *71*, *81*, *82*, *88*, *89*, *91*, 93, *96*, *97*, *99*
minFormula,mineral-method (mineral-class), 89
minFormulaAll (minFormula), 93
minMain, *72*, *81*, *82*, *84*, *85*, *89*, *91*, *95*, 96, *99*
minValues, *72*, *81*, *82*, *84*, *85*, *88–91*, *95–97*, 98
minValues,mineral-method (mineral-class), 89
minValuesAll (minValues), 98
Molecular weights, 99
molecularWeight, *27*, *104*, *105*, *124*
molecularWeight (Molecular weights), 99
multiple, *5*, *115*
multiple (Multiple plots), 100
Multiple plots, 100
multipleMjr (Multiple plots), 100
multiplePerPage (Plate), 114
multipleTrc (Multiple plots), 100
MW (Molecular weights), 99

normalize2total (recast), 135

olComp, *65*, *86*
olComp (minComp), 85
olivine, 103
optimize, *33*, *35*
options, *67–69*, *73*, *154*
oxide2oxide, 104, *105*, *124*
oxide2ppm, *104*, 105, *124*

pairs, *106*, *108*
pairsCorr, 105
pairsMjr (pairsCorr), 105
pairsTrc (pairsCorr), 105
panel.cor (pairsCorr), 105
panel.cov (pairsCorr), 105
panel.hist (pairsCorr), 105
panel.smooth (pairsCorr), 105
par, *7*, *48*, *50*, *57*, *60*, *61*, *117*, *128*, *178*, *180*
pdf, *108*
pdfAll, 108, *131*
peekDataset, *109*, 109, *122*, *123*, *132*
peterplot, *5*, 110
phasePropPlot, 112
Plate, *5*, *55*, *85*, *102*, *103*, 114, *117*, *119*

Plate editing, *5*, *55*, *102*, *103*, *115*, 116, *119*
plate0YLim *(*Plate editing*)*, 116
plateAnnotationsRemove *(*Plate editing*)*, 116
plateBW *(*Plate editing*)*, 116
plateCex *(*Plate editing*)*, 116
plateCexLab *(*Plate editing*)*, 116
plateCexMain *(*Plate editing*)*, 116
plateCol *(*Plate editing*)*, 116
plateExpand *(*Plate editing*)*, 116
plateExtract *(*Plate editing*)*, 116
plateLabelSlots, 118
plateLoad *(*Plate*)*, 114
platePch *(*Plate editing*)*, 116
platePS *(*Plate*)*, 114
plateRedraw *(*Plate*)*, 114
plateSave *(*Plate*)*, 114
plateUser, *5*, *61*
plateUser *(*Plate editing*)*, 116
plateXLim *(*Plate editing*)*, 116
plateYLim *(*Plate editing*)*, 116
plComp, *65*, *86*
plComp *(*minComp*)*, 85
plot, *19*, *21*, *115*, *173*
plot.default, *49*, *56*, *172*
plotWithCircles, *5*, 120, *150*, *164*
plotWithLimits *(*binary*)*, 18
points, *57*
pokeDataset, *76*, *77*, *109*, *122*, 122, *131*, *132*
polygon, *32–35*
postscript, *131*
ppm2oxide, *104*, *105*, 123
pr.comp.clr, *125*
pr.comp.clr *(*clr.transform*)*, 27
prComp, *29*, 124
princomp, *29*, *125*
print,mineral-method *(*mineral-class*)*, 89
print,mineral.db-method *(*mineral.db-class*)*, 92
printSamples, 125
printSingle, 127
profiler, *5*, 128
psAll, *108*, 130
purgeDatasets, *109*, *122*, *123*, 131
PxPapike *(*minFormula*)*, 93
PxPapike,mineral-method *(*mineral-class*)*, 89
pyroxene, 132

quantile, *14*
quit, *133*
quitGCDkit, 133

r2clipboard, 133
read.table, *77*, *138*
recalcOptions, 134
recast, 135
refreshFig *(*figRedraw*)*, 58
regex, *23*, *136*, *152*
regular expression, *151*
regular expression *(*Regular expressions*)*, 136
Regular expressions, 136
regular expressions, *144*
regular expressions *(*Regular expressions*)*, 136
regular.expression *(*Regular expressions*)*, 136
regular.expressions, *23*, *147*
regular.expressions *(*Regular expressions*)*, 136

sampleDataset, 137
saveData, *77*, *78*, 138
saveResults, 139
sazava, 139
Scale: axis labels *(*figScale*)*, 59
Scale: subtitle *(*figScale*)*, 59
Scale: symbols *(*figScale*)*, 59
Scale: title *(*figScale*)*, 59
scattersmooth, 140
selectAll, 143, *152*
selectByLabel, *144*, 144, *152*
selectByMineral, 145
selectColumnLabel, *12*, *13*, *15*, *16*, *19*, *20*, *25*, *37*, *39*, *70*, *121*, *127*, *142*, *144*, 146, *161*, *163*, *169*, *170*, *172*, *173*
selectColumnsLabels, *30*, *106*, *124*, *126*, *135*, *147*, 147, *165*, *167*
selectDataset, *122*, *123*, *132*
selectDataset *(*peekDataset*)*, 109
selectPalette, *12*, *13*, *64*, 149
selectSamples, *30*, *106*, *124*, *126*, *155*, *165*, *167*, *169*
selectSamples *(*selectSubset*)*, 151
selectSubset, *19*, *20*, *22*, *51*, *121*, *129*, *143*, *144*, 151, *164*, *173*
setCex, *67*, *69*, 153
setShutUp, 154
setTransparency, 155
setwd, *77*
showColours, *10–12*, *14–16*, *48*, *50*, *52*, *76*, *77*, *117*, *138*, *150*, *155*, 156, *158*
showColours2 *(*showColours*)*, 156
showLegend, *10–12*, *15*, *16*, *47*, *48*, *50*, 157

showSymbols, *10–12*, *15, 16*, *76, 77*, *117*, *138*, *158*, 159
statistics, *160*, *166*, *169, 170*
Statistics: Correlation: majors (pairsCorr), 105
Statistics: Correlation: traces (pairsCorr), 105
Statistics: Majors summaryAll/selection (summaryAll), 165
Statistics: Majors summaryByGroups (summaryByGroup), 166
Statistics: Trace summaryAll/selection (summaryAll), 165
Statistics: Trace summaryByGroups (summaryByGroup), 166
statsByGroup, 159
statsByGroupPlot, 160
strip, 161, *164*
stripBoxplot, *162*, 162
stripplot, *161, 162*, *164*
Subset by range, 164
subsetBoolean, *151*
subsetBoolean (Boolean conditions), 22
subsetRange, *151*
subsetRange (Subset by range), 164
summaryAll, *160*, 165, *170*
summaryByGroup, *160*, *166*, 166, *170*
summaryByGroupMjr (summaryByGroup), 166
summaryByGroupTrc (summaryByGroup), 166
summaryMajor (summaryAll), 165
summaryRangesByGroup (summaryByGroup), 166
summarySingle, *160*, *166*, 168, *170*
summarySingleByGroup, *160*, *166*, *170*, 170

ternary, *5*, *55*, *115*, 171
threeD, *5*, 173
tk_winDialog, 176, *177*
tk_winDialogString, *176*, 177
tkentry, *177*
tkmessageBox, *176*
tkSelectVariable, 175
trendTicks, 178
triplot (ternary), 171
triplotadd, *57*
triplotadd (ternary), 171

webMineral, 181
wholeRock, 182
winDialog, *176*
winDialogString, *177*
write.dbf, *41*

Zooming: Scale x axis (figZoom), 61
Zooming: Scale y axis (figZoom), 61
Zooming: Zoom in (figZoom), 61
Zooming: Zoom out to original size (figZoom), 61