Visualization of three- and four-part (sub)compositions with R

M. BREN[1,3] & V. BATAGELJ[2,3]

[1]*University of Maribor, Faculty of Organizational Sciences, Kidričeva 55a, 4000 Kranj, Slovenia (e-mail: matevz.bren@fov.uni-mb.si)*

[2]*University of Ljubljana, Faculty of Mathematics and Physics, Jadranska 19, 1000 Ljubljana, Slovenia*

[3]*Institute of Mathematics, Physics and Mechanics, Jadranska 19, 1000 Ljubljana, Slovenia*

Abstract: In 2003 the MixeR (Mixtures with R) project was started and work began to develop a library of functions written in R to support the analysis of compositional data, i.e. mixtures. This paper presents the 'mix' object in R, reading different data file formats and some MixeR routines for graphical presentation of three- and four-part (sub)compositions in ternary diagrams and tetrahedrons. Additional graphical features and use of parameters are applied on real data – a glacial dataset and dataset of the researcher's daily activities, both from Aitchison's (1986) book. All these routines and datasets are available at http://vlado.fmf.uni-lj.si/pub/MixeR

The paper begins by introducing the two datasets – data on *Researcher's daily activities* and the *Glacial data* that will serve as examples. Reader are strongly recommended to sit in front of the computer with R installed (R beginners – see Downloading and installing R in Boogaart & Tolosana-Delgado 2006), typing the examples outlined here. With this the reader will also see all figures produced in colour and will understand the ideas on classification and classes more clearly. All MixeR routines and the two datasets are available at http://vlado.fmf.uni-lj.si/pub/MixeR

Researcher's daily activities data

The dataset No. 31 from Aitchison (1986) gives activity patterns of a statistician for 20 days. The proportions of a day spent teaching, in consultation, administrative work, research, other wakeful activities and sleep are given.

Data show the proportions of the 24 hours devoted to each activity, recorded on each of the 20 days. The activity proportions, not the values in hours are given. Therefore, the data are portions of a day summing to one, thus compositional, i.e. mixtures.

The data are stored in a file in matrix form, days as rows and activities as columns and the first row comprising the abbreviations of the activities, i.e. variable names: **teac** – teaching, **cons** – consultation, **admi** – administration, **rese** – research, **wake** – other wakeful activities and **slee** – sleep are given. This dataset is presented in Table 1.

The six activities may be divided into two categories: 'work' comprising activities 1, 2, 3, 4, and 'leisure' comprising activities 5 and 6. These data will be used to present the subcompositional concepts, visualization of the data in ternary diagrams presenting variability with border percentile lines, centring, etc.

Glacial dataset

From Aitchison (1986) dataset No. 18 gives 92 samples of pebbles of glacial tills sorted into four categories: red sandstone, grey sandstone, crystalline and miscellaneous. The percentages by weight of these four categories and the total pebbles counts are recorded. These data are stored in the 'CoDa' data file – each line comprising just one record: in the first line the data file name, in the second the number of variables, in the third the number of cases, in the next rows the variable labels and then No. 1 for the first case and the variable values for the first case, then No. 2 and the variables values for the second case, etc. until the last, No. 92 (Table 2).

These data in Table 2 will be used to present the visualization in tetrahedrons and KiNG Mage viewer animation, dealing with zeros, Aitchison's distance computation and classification.

Compositional data analysis software tools

First, a brief, not exhaustive history, of software tools for compositional data analysis will be given.

CoDa, a microcomputer package for the statistical analysis of compositional data was the first

From: Buccianti, A., Mateu-Figueras, G. & Pawlowsky-Glahn, V. (eds) *Compositional Data Analysis in the Geosciences: From Theory to Practice*. Geological Society, London, Special Publications, **264**, 129–143.
0305-8719/06/$15.00 © The Geological Society of London 2006.

Table 1. *Researcher's daily activities dataset*

	teac	cons	admi	rese	wake	slee
1	0.162	0.041	0.138	0.123	0.254	0.282
2	0.200	0.039	0.073	0.076	0.346	0.266
3	0.201	0.082	0.115	0.146	0.194	0.261
4	0.134	0.077	0.107	0.146	0.214	0.321
5	0.224	0.080	0.091	0.162	0.195	0.248
6	0.144	0.063	0.103	0.123	0.316	0.252
7	0.125	0.054	0.137	0.102	0.312	0.270
8	0.127	0.077	0.110	0.101	0.341	0.244
9	0.139	0.052	0.128	0.111	0.266	0.304
10	0.108	0.052	0.082	0.075	0.413	0.270
11	0.187	0.091	0.113	0.116	0.264	0.228
12	0.184	0.070	0.066	0.151	0.305	0.216
13	0.155	0.086	0.101	0.119	0.225	0.315
14	0.181	0.097	0.081	0.164	0.271	0.206
15	0.224	0.096	0.101	0.142	0.203	0.234
16	0.198	0.067	0.139	0.154	0.162	0.281
17	0.214	0.073	0.102	0.130	0.201	0.281
18	0.132	0.037	0.148	0.099	0.307	0.277
19	0.167	0.073	0.127	0.122	0.266	0.245
20	0.166	0.064	0.101	0.145	0.242	0.282

From dataset 31 (Aitchison 1986).

software on compositions, written in Quick Basic by John Aitchison and available with his book (Aitchison 1986). CoDa was later upgraded by John Bacon-Shone.

Table 2. *Gacial dataset*

b:glacial.dat
5
92
Case no
A
B
C
D
Count
1
91.8
7.1
1.1
0
282
2
88.9
10.1
0.5
0.5
368
...
92
31.4
65.9
2.7
0
698

From dataset 18 (Aitchison 1986).

CoDaPack freeware software was next, written in Excel in 2001 by Santiago Thió Fernandez de Henestrosa and Josep Antoni Martín-Fernández from Girona compositional research group. An introduction is available in this volume (Thió-Henestrosa & Martín-Fernández 2006).

There were also some attempts written in R – a language and environment for statistical computing and graphics. R (http://www.r-project.org/), is 'GNU S'; it provides a wide variety of statistical and graphical techniques (linear and non-linear modelling, statistical tests, time-series analysis, classification, clustering, etc.). Further extensions can be provided as packages.

Basic Compositional Data Analysis functions for S+/R comprising basic operations, transformations, estimators and plots, written by Joel Reynolds & Dean Billheimer (2002) from the Washington University are available at http://www.biostat.wustl.edu/archives/html/s-news/2003-12/msg00139.html

In 2003 work began to develop a library MixeR of functions in R to support the analysis of compositional data, i.e. mixtures (Bren & Batagelj 2003) Routines were provided for:

- **Operations on compositions:** perturbation and power transformation, subcomposition with or without residuals, centring of the data, computing Aitchison's, Euclidean, Bhattacharyya distances and compositional Kullback–Leibler divergence – see Martín-Fernández *et al.* (1999).
- **Graphical presentation** of three- and four-part (sub)compositions in ternary diagrams and

tetrahedrons with additional features: geometric mean of the dataset, the percentiles and ratio lines, centring of the data, notation of individual data in the set, marking and colouring of subsets of the dataset, their geometric means, etc.

- **Log-ratio transformations** of compositions into real vectors that are amenable to standard multivariate statistical analysis, etc.

The current version of the MixeR library is available at http://vlado.fmf.uni-lj.si/pub/MixeR

In April 2005 Kjetil Halvorsen, author of the 'Fahrmeir' R package, reported his work on coding some compositional routines in R (operations on compositions, alr and clr transformations, etc.).

In June 2005 a 'compositions' package, written by K. Gerald van der Boogaart and Raimon Tolosana-Delgado was published and is now available at http://cran.r-project.org/src/contrib/Descriptions/compositions.html. To analyse compositions this package supports four different multivariate scales represented by four different classes: '`rplus`' – the total amount is meaningful and data are analysed in real geometry; '`rcomp`' – the total amount is meaningless or the individual amounts are parts of a whole in equal units and data are analysed in real geometry; '`acomp`' – the total amount is meaningless or the individual amounts are parts of a whole in equal units and the data should be analysed in a relative, i.e. Aitchison's geometry; '`aplus`' – the total amount is meaningful and the data should be analysed in relative geometry. Choosing the right type of analysis according to the data is left to the user. The package manual 'The compositions Package' and the introduction 'Using the R package "compositions"' are also available. An introduction to this package is also available in this volume (Boogaart & Tolosana-Delgado 2006).

The mixture class in R

The mix object in R will be presented, dealing with different data file formats and some R routines for graphical presentation of three- and four-part (sub)-compositions in ternary diagrams and tetrahedrons, not incorporated in the 'compositions' package. Additional features will be applied: plotting the geometric mean of the dataset, ratio lines and/or percentile lines, marking and colouring subsets of the dataset and centring of the dataset.

The input mixture data consist of a data matrix preceded by a title. They are represented as an R 'data frame', an object `m` consisting of

`m$tit`	the title of the dataset,
`m$sum`	the value of the row sums, if constant,
`m$sta`	the status of the mix object with values
−2	matrix contains negative elements,
−1	zero row sum exists,
0	matrix contains zero elements,
1	matrix contains positive elements, rows with different row sum(s),
2	matrix with constant row sum and
3	normalized mixture, the row sums are all equal to 1.
`m$mat`	the matrix with the data, and
`m$class`	the special attribute of the object, used to allow for an object-orientated style of programming in R. For the MixeR purpose the class is defined 'mixture'.

Example

of the mixture object – the dataset of researcher's daily activities (Table 1). Proportions of a day in activity are given for a statistician for 20 days.

```
> m <- mix.Read('activity.dat')
> m
```

Here the R cursor mark '>' denotes the start of the command line and '<-' is an assignment operator. The output is

```
$tit
[1] ''Researcher's daily activities''
$sum
[1] NA
$sta
[1] 1
$mat
   teac  cons  admi  rese  wake  slee
1  0.162 0.041 0.138 0.123 0.254 0.282
2  0.200 0.039 0.073 0.076 0.346 0.266
3  0.201 0.082 0.115 0.146 0.194 0.261
...  ...    ...    ...
19 0.167 0.073 0.127 0.122 0.266 0.245
20 0.166 0.064 0.101 0.145 0.242 0.282
attr (,''class'')
[1] ''mixture''
```

It should be explained that the `$sum` is `NA`, not available, because row sums are not all exactly equal to one due to the rounding errors; therefore, the status `$sta` is one, i.e. the matrix contains positive elements and rows with different row sums.

The 'mix' procedures in R

Some basic MixeR routines are presented.

```
mix.Read(file, eps=1e-6)
```

Reads a mixture data from the file and returns it as a mixture object. If |m$sum-1|<eps it sets m$sta=3. The default value for eps is 1e-6.

```
mix.ReadML(file, eps=1e-6)
```

Reads a 'CoDa' data file and returns a mixture object. If |m$sum-1|<eps it sets m$sta=3. The default value for eps is 1e-6.

```
mix.Check(m, eps=1e-6)
```

Determines the m$sum and m$sta of a given mixture object m. The default value for eps is 1e-6.

```
mix.Normalize(m, c=1)
```

Normalizes a given mixture object m if m$sta>=0. The rows sums are now normalized to the constant c with default value c=1.

```
mix.Matrix(a, t)
```

Joins a matrix data a and the title t into a mixture object.

```
mix.Random(nr, nc, c=1)
```

Constructs a random mix object with nr rows and nc columns and constant row sum c with default value c=1. The command matrix (runif(nr*nc),nr,nc)) is applied to calculate matrix elements where runif (n, min=0, max=1) function generates random deviates of uniform distribution. Then the row sums are normalized to the constant c.

```
mix.Sub(m, k, Normalize=TRUE)
```

Output mix object is computed as a subcomposition of m without the columns given by the components of the vector k. The output mix object is normalized if Normalize=TRUE, the default value.

```
mix.SubRes(m, k)
```

Output is the normalized subcomposition without the columns given by the components of the vector k and amalgamated in the residual.

```
mix.Extract(m, k, Normalize=TRUE)
```

Gives the subcomposition of m with only the columns given by components of the vector k, normalized if Normalize=TRUE, the default value.

```
mix.ExtractRes(m, k)
```

Gives the subcomposition with the columns given by the components of the vector k; all the rest is amalgamated in the residual. Output is the normalized mixture object with length(k)+1 columns.

The subcomposition routines

Example

Determine the three-part subcompositions of activity data comprising teaching, consulting and research activities, therefore excluding the 3rd, 5th and 6th columns. To define a vector in R the c – concatenate command is applied.

```
> m <- mix.Read('activity.dat')
> mix.Sub(m, c(3,5,6))

$tit
[1] ''Researcher's daily
      activities''

$sum
[1] 1

$sta
[1] 3

$mat
   teac   cons   rese
1  0.497  0.126  0.377
2  0.635  0.124  0.241
...          ...    ...
20 0.443 0.171  0.387

attr(,''class'')
[1] ''mixture''
```

Example

To determine the three-part subcompositions of activity data on teaching and research activities, with all the rest (the 2nd, 3rd, 5th and 6th variables) amalgamated in the residual.

```
> mix.ExtractRes(m,  c(1,4))

$tit
[1] ''Researcher's daily
      activities''

$sum
[1] 1

$sta
[1] 3
```

```
$mat
     teac   rese   residual
  1  0.162  0.123   0.715
  2  0.200  0.076   0.724
  ...        ...     ...
  20 0.166  0.145   0.689


  attr(,``class'')
  [1] ``mixture''
```

Visualization in the ternary diagram routine

The `mix.Ternary` routine draws a ternary diagram with points marking the data. The routine `mix.Ternary(m, dist, distG, cls, Centre, Borders, Gmean)` has the following parameters:

`m`	the mix object,
`dist`	displaces the numbers marking percentile lines for additional space given by the components of the vector `dist`. First component for percentile lines to the vertex No. 1 = top, second to the vertex No. 2 = right, and third to the vertex No. 3 = left, i.e. each component corresponds to one vertex. This displacing is needed to prevent overlaying. The default value is `dist=c(0.05, 0.05, 0.05)`
`distG`	displaces the numbers marking percentile lines of the geometric mean for additional space to prevent overlaying. Additional space is given by the components of the vector `dist`, each component corresponds to one vertex. The default value is `distG=c(0.05, 0.05, 0.05)`.
`cls`	colours of the percentile lines.
`Centre`	centres the dataset if `Centre=TRUE`,
`Borders`	draws border percentile lines if `Borders=TRUE`,
`Gmean`	draws the geometric mean of the data if `Gmean=TRUE`.
	The default value for all these options is `FALSE`.

Example

In Figure 1, the plots of activity data subcompositions in ternary diagrams with the geometric mean and the border percentile lines are produced by the following commands

```
> mix.Ternary(mix.Sub(m, c(3,5,6)),
    distG=c(.1,.1,.1), Gmean=T)
> mix.Ternary(mix.Sub(m, c(3,5,6)),
    Borders=T, cls=c(``red'',
    ``magenta'', ``blue''))
```

In Figure 2, the plots of activity data subcompositions (with residual) in ternary diagrams with border percentile lines are produced by the following commands

```
> mix.Ternary(mix.SubRes
    (m, c(2,3,5,6)),
    dist=c(.05,.1,.05), Borders=T)
> mix.Ternary(mix.SubRes
    (m, c(2,3,5,6)),
    dist=c(.05,.1,.05),
    Borders=T, Centre=T)
```

Example

The glacial dataset (Table 2) consists of percentages by weight for 92 samples of pebbles of glacial tills sorted into four categories – red sandstone, grey sandstone, crystalline and miscellaneous. The percentages by weight of these four categories and the total pebbles counts are recorded. The data are stored in a CoDa data file format.

```
> m <- mix.ReadML('glacial.dat')
> m

$tit
[1] ``GLACIAL DATA 92 samples of
      pebbles of glacial tills
      sorted into four categories
      percentages by weight''

$sum
[1] NA

$sta
[1] 0

$mat
    A     B     C    D    Count
1   91.8   7.1  1.1  0.0  282
2   88.9  10.1  0.5  0.5  368
...  ...   ...   ...  ...  ...
90  15.9  83.3  0.8  0.0  245
91  16.9  74.3  1.2  5.9  575
92  31.4  65.9  2.7  0.0  698
```

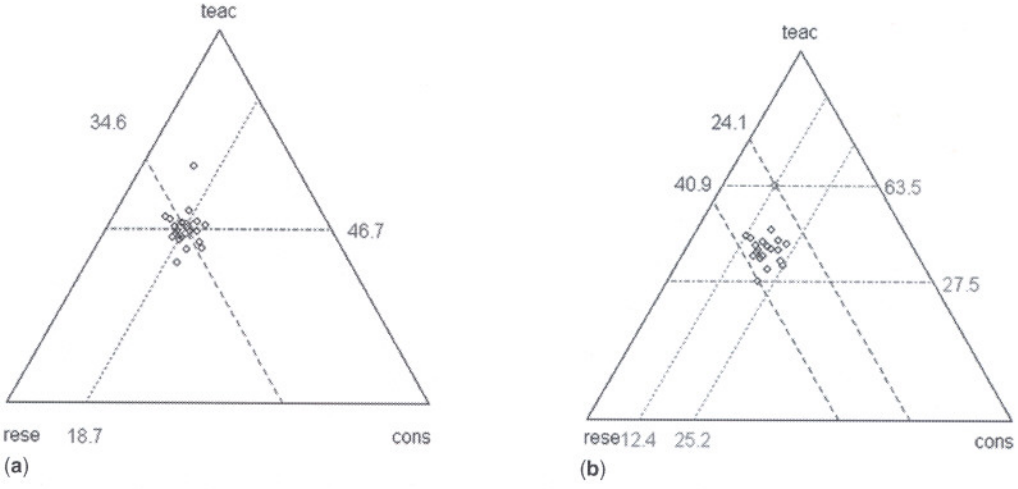

Fig. 1. Three-part (teaching, consultation and research) subcompositions with (**a**) the geometric mean; and (**b**) border percentile lines.

```
attr(,``class'')
[1] ``mixture''
```

```
> mix.Ternary(mix.Sub(m,c(4,5),
    Gmean=T)
> mix.Ternary(mix.Sub(m,c(4,5),
  Borders=T, Centre=T)
```

See Figure 3.

The percentile lines routine

The routine that draws percentile lines into a drawn ternary diagram is `percentile.lines(y, direction, cls, dist, lt)` with parameters

`y`	the vector of percents or decimal values of percentile lines;
`direction`	directions for percentile lines with values 1 – percentile lines to the vertex No. 1 = top, 2 – percentile lines to the vertex No. 2 = right, and 3 – percentile lines to the vertex No. 3 = left. The default value is `direction=1:3` i.e. all directions;
`cls`	the vector with colours, each component corresponds to one vertex. The default value is `cls=c(``yellow'', ``yellow2'', ``yellow3'')` visible on the screen but for printing or presentation stronger colours are advised;
`dist`	moves the numbers marking the percentile lines for additional space given by the components of the vector `dist`, to prevent overlaying. The default value of `dist=c(0.05, 0.05, 0.05);`
`lt`	is the vector with line types, the `lty` parameter in R graphics routines (values 1, 2, . . . , 10), each component corresponds to one vertex. The default value is `lt=c(4,3,2)`.

Example

A normalized mix object `m` with nine cases and three variables, i.e. 9×3 matrix, is constructed, having 0.1 to 0.9 values in the first column, ratios of one half between the second and third. A ternary diagram is drawn with these nine points in different colours – `cls`, different shapes – `pch` and the size `cex=1` (see Fig. 4a).

```
$tit
[1] ``Deciles values in the first
      column''

$sum
[1] 1

$sta
[1] 3
```

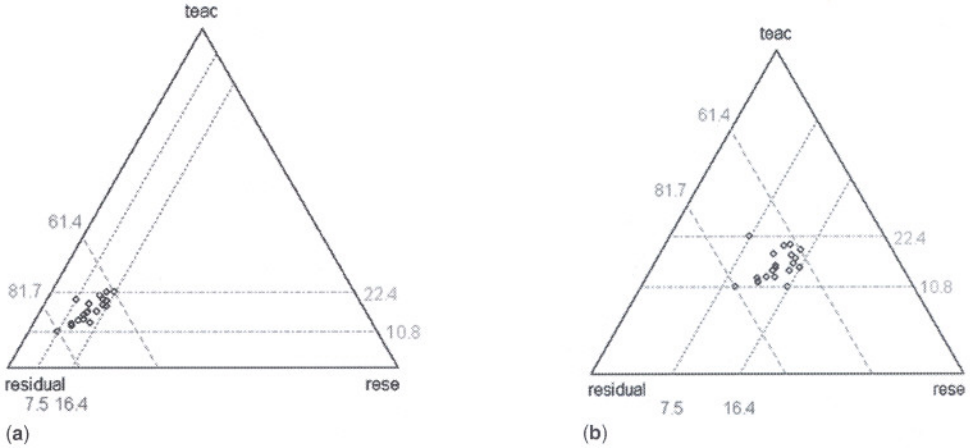

Fig. 2. Three-part (teaching, research and residual) subcompositions with (**a**) borders percentile lines; and (**b**) centred for better visualization of the differences between cases. To avoid misunderstanding of this centred visualization, borders percentile lines with exact max variation values are obligatory.

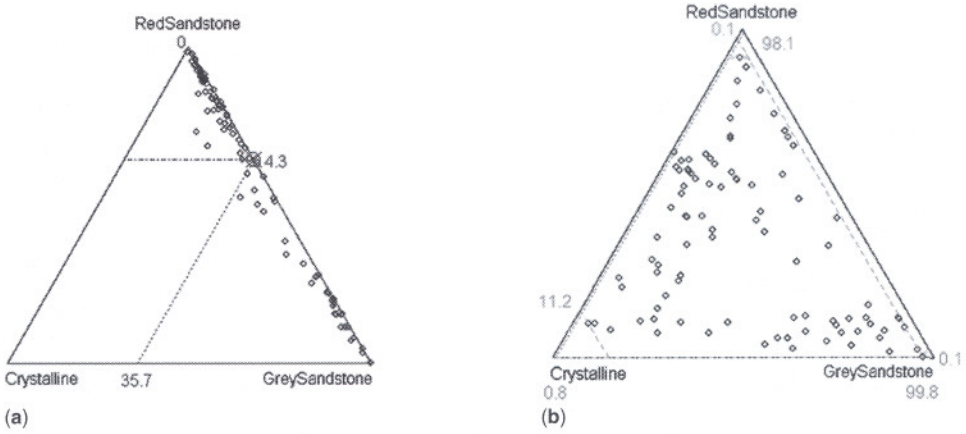

Fig. 3. Three-part (red sandstone, grey sandstone and crystalline) subcompositions, with (**a**) geometric mean; and (**b**) centred for better visualization of the differences between cases – border percentile lines showing actual variation.

```
$mat
   aa         bb         cc
1 0.1 0.30000000 0.60000000
2 0.2 0.26666670 0.53333330
3 0.3 0.23333330 0.46666670
4 0.4 0.20000000 0.40000000
5 0.5 0.16666670 0.33333330
6 0.6 0.13333330 0.26666670
7 0.7 0.10000000 0.20000000
8 0.8 0.06666667 0.13333330
9 0.9 0.03333333 0.06666667

attr(,``class'')
[1] ``mixture''

> cls <- c(``khaki'', ``pink'',
    ``sienna'',``plum'',``orchid'',
    ``violet'', ``tomato'', ``tan'',
    ``purple'')
> mix.Ternary(m, col=cls, pch=0:8,
    cex=1)
> perc.lines(10*1:9,  dir=1,
    cls=``cyan'', lt=1)
```

To draw the ternary diagram in Figure 4b, use the

```
mix.Random(nr, nc, s=1)
```

routine is used that constructs a random mix object with `nr` rows and `nc` columns with a constant row sum `s`.

```
> mix.Ternary(mix.Random(22,3))
```

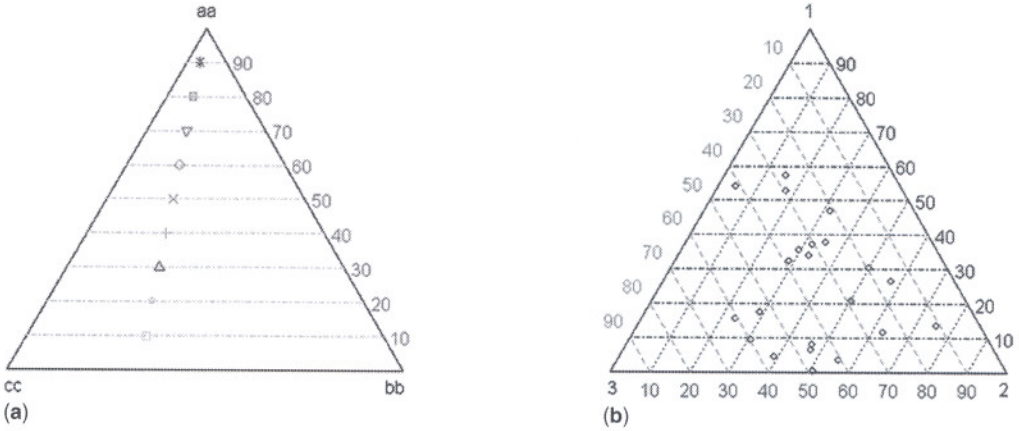

Fig. 4. Three-part compositions: (**a**) deciles values in the first column, constant ratios half between the second and the third column plotted in the ternary diagram with deciles lines in the first direction; (**b**) ternary diagram with the random 22 points and deciles lines in all three directions.

```
> perc.lines(10*1:9,
    cls=c(``blue'', ``blueviolet'',
    ``violet''))
```

The ratio lines routine

The command that draws lines of constant ratios of two components into a drawn ternary diagram is `ratio.lines(y, direction, cls, dist)`. The routine parameters are

`y`	the vector of ratios for ratio lines;
`direction`	directions for ratio lines with value 1 – ratio lines to the vertex No. 1, i.e. $x2/x3 = y$, 2 – ratio lines to the vertex No. 2, i.e. $x1/x3 = y$, and 3 – ratio lines to the vertex No. 3, i.e. $x1/x2 = y$. The default value is `direction=1:3` that stands for all the directions;
`cls`	the vector with colours, each component corresponds to one vertex. The default value is `cls=c(``green'', ``green3'', ``yellowgreen'')` visible on the screen but for printing or presentation stronger colours are advised,
`dist`	moves the numbers marking ratio lines for additional space given by the components of the vector `dist`, to prevent overlaying. The default value is `dist=c(0.05, 0.05, 0.05)`.

Example

A matrix with nine cases and three variables is constructed, first triple of cases having a constant ratio of one half between second and third variables, second triple between first and third ... and each triplet is coloured (see Fig. 5a). In this ternary diagram we draw the 1/7, 1/3, 1/2, 1, 2, 3 and 4, ratio lines to all three sides (see Fig. 5b).

```
m <- matrix(c(7, 4, 1, 1, 2, 3, 1,
    2, 3, 1, 2, 3, 7, 4, 1, 2, 4, 6,
    2, 4, 6, 2, 4, 6, 7, 4, 1),
    nrow=9)
m <- mix.Normalize(mix.Matrix(m,
    'Constant ratios'))
> m
$tit
[1] ``Constant ratios''

$sum
[1] 1

$sta
[1] 3

$mat
     [,1] [,2] [,3]
[1,]  0.7  0.1  0.2
[2,]  0.4  0.2  0.4
[3,]  0.1  0.3  0.6
[4,]  0.1  0.7  0.2
[5,]  0.2  0.4  0.4
[6,]  0.3  0.1  0.6
[7,]  0.1  0.2  0.7
[8,]  0.2  0.4  0.4
[9,]  0.3  0.6  0.1
$class
```

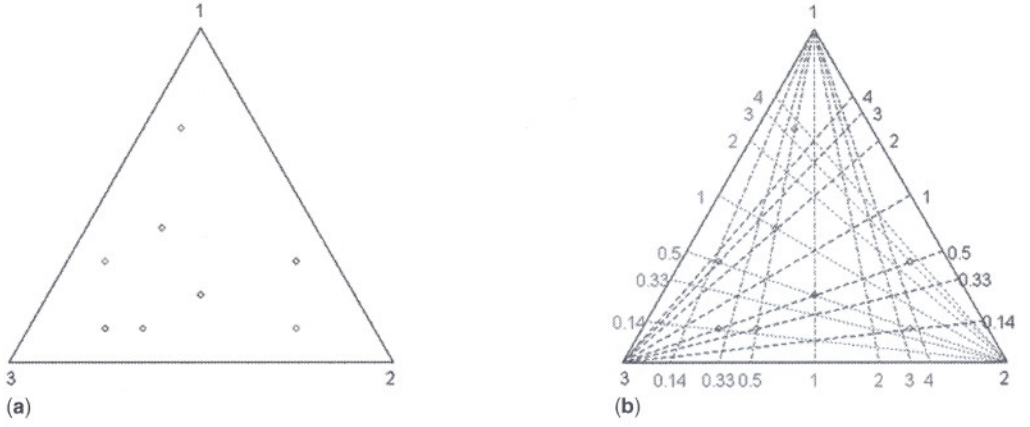

Fig. 5. Three-part compositions with (**a**) constant ratios of all combinations of two components, and (**b**) with the constant ratio lines also drawn for ratios 1/7, 1/3, 1/2, 1, 2, 3 and 4 in all three directions.

```
  [1] ``mixture''
>t<-c(rep(1,3),rep(2,3),rep(3,3))
>co <- c(``red'',``magenta'',
  ``blue'') # different colours
  of points
>mix.Ternary(m, col=co[t]) # draws
  ternary diagram
>ratio.lines
  (c(1/3,1/2,1,2,3,4, 1/7),
  cls=co)# draws ratio lines
```

Visualization with the tetrahedron routine

The `mix.Q2kin` routine transforms a four-part mixture `m` into three-dimensional XYZ coordinates using quadrays transformations and saves them as a file.kin. This transformation applies Quadrays and XYZ by K. Urner and Quadray formulas by T. Ace available on the web. The kin file is displayed as 3D animation using KiNG or MAGE viewer—free software available at http://kinemage.biochem.duke.edu.

The `mix.Q2kin (kinfile, m, clu=NULL, vec=NULL, king=TRUE, scale=0.2, col=1)` routine's parameters are

`kinfile`	the name of a `file.kin`,
`m`	the mix object with four variables,
`clu`	partition determining the colours of points,
`vec`	vector of values determining points sizes,
`king`	FALSE for Mage, TRUE for King,
`scale`	relative size of points, and
`col`	colour of points if `clu=NULL`.

Example

From the activity data mix object `m` will be constructed. A four-part composition with variables teaching, consulting and research, administration work, and leisure comprising other wakeful activities and sleep. The Aitchison distance will be computed and the complete linkage classification method performed. Out of the dendrogram four clusters will be detected and drawn with the `mix. Q2kin` command in a `ac4.kin` file to be displayed with the KiNG viewer in a tetrahedron using different colours (Figs 6 and 7).

```
> m$mat<-cbind(m$mat[,1],
    m$mat[,2]+m$mat[,4],m$mat[,3],
    m$mat[,5]+m$mat[,6])

> dimnames(m$mat)[[2]]<-
    c('teach','cons&rese','admi',
    'leas')
> m

$tit
[1] ``Researcher's daily activi-
      ties''

$sum
[1] NA

$sta
[1] 1

$mat
   teach cons&rese admi   leas
1  0.162  0.164     0.138  0.398
2  0.200  0.115     0.073  0.490
...  ...    ...       ...    ...
19 0.167  0.195     0.127  0.410
20 0.166  0.209     0.101  0.386
```

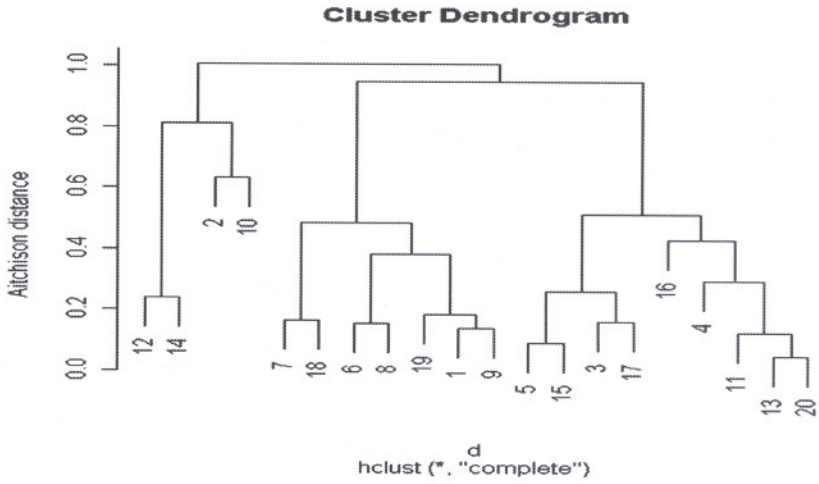

Fig. 6. Activity dataset classification represented by a dendrogram.

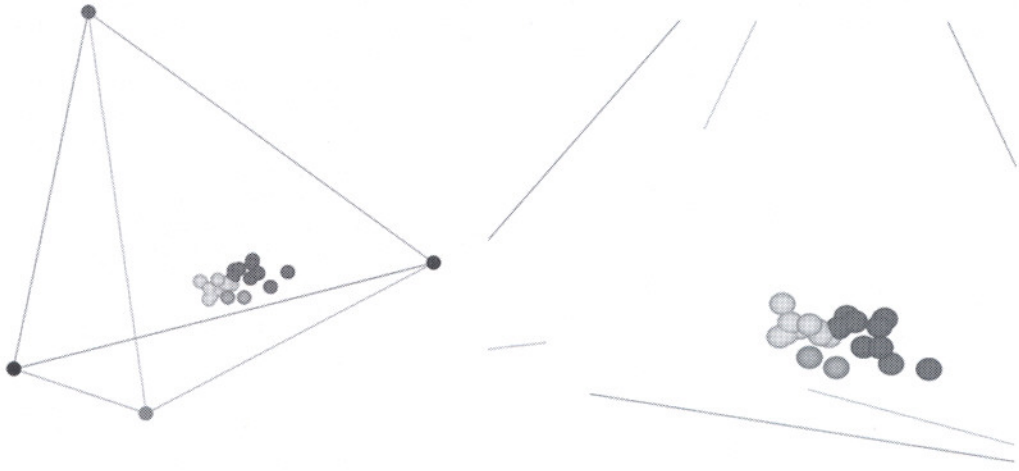

Fig. 7. Two snapshots of 3D KiNG view of tetrahedral display of the activity data – four-part compositions.

```
> d<-mix.Ait(m) # computes a matrix
    of Aitchison's distances
    between cases
> hc<-hclust(d,method=``complete'',
    members=NULL) # performs the
    complete linkage classification
    method
> plot(hc, labels=NULL, hang=0.1,
    main=``Cluster Dendrogram'',
    ylab= ``Aitchison distance'')
> mix.Quad2kin(`ac4.kin', m,
    clu=cutree(hc,4), scale=0.2)
```

The kinemage is a dynamic, 3D layout. One can take advantage of that:

- by rotating it and twisting it around with the mouse click near the centre of the graphics window and slowly dragging right or left, up or down;
- by clicking on points with the mouse, the label associated with each point will appear in the bottom left of the graphics area;
- also the (Euclidean) distance from this point to the previous will be displayed;
- the right button drag can be used to zoom in and out of the picture.

This layout also supports colouring and different sizing of points.

Zero values replacement routines

The routine that checks data for 'zero' and negative values is `mix.CheckData (m, eps=1e-6, Detect=FALSE)` with parameters

`m`	the mix object,
`eps`	turning point value for zero checking, and
`Detect`	if `TRUE` output is also the mix object with matrix elements equal −1 at negative data values, 0 at zero i.e. $\lvert$`m$[i,j]`$\rvert$ < `eps` values, and 1 at all positive values.

Output is the column number `-i`, the row number `-j`, the data value and the consecutive number of negative and/or zero values. If there is no negative and/or 'zero' values, output is `''No negative, no zero values in the data''`.

Example

Here is a mix object `m` with eight cases, negative values in first column and 0.1 to 1e-09 in the last.

```
$tit
[1] ''Negative values in first
      column, 0.1 to 1e-09 in the
      last.''

$sum
[1] NA

$sta
[1] -2

$mat
   aa  bb  cc  dd  ee   ff
1  -3   3   3   3   3   1e-01
2  -3   3   3   3   3   1e-02
3  -3   3   3   3   3   1e-03
4  -3   3   3   3   3   1e-04
5  -3   3   3   3   3   1e-05
6  -3   3   3   3   3   1e-06
7  -3   3   3   3   3   1e-07
8  -3   3   3   3   3   1e-08
9  -3   3   3   3   3   1e-09

$class
[1] ''mixture''

> mix.CheckData(m, Detect=F)
negative values in the data
negative value column 1 row 1 value
  -3 No. 1
negative value column 1 row 2 value
  -3 No. 2
...    ...    ...
negative value column 1 row 8 value
  -3 No. 8
negative value column 1 row 9 value
  -3 No. 9
zero values in the data
zero value column 6 row 7 value 1e-
  07 No. 1
zero value column 6 row 8 value 1e-
  08 No. 2
zero value column 6 row 9 value 1e-
  09 No. 3
```

The `mix.ZeroReplaceSimp` routine replaces all the 'zero' (i.e. less than eps) values applying the Simple replacement strategy – see Martin-Fernandez *et al.* (2003). Output is a mix object with all 'zero' values replaced and the row sums preserved.

In the routine `mix.ZeroReplaceSimp(m, de, Col=TRUE, eps=1e-6)` the parameters are

`m`	the mix object,
`de`	vector of imputed values,
`eps`	value for zero checking, and
`Col`	if `TRUE` one component of `de` is imputed in each column. If `de` length is less than the number of columns i.e. variables, the `de` is repeated to adequate length. If `Col=FALSE`, one component of `de` is imputed in each row with zero components, no length adjustment is made. The default value for `Col=TRUE`.

Example

Take a mix object `m` with five 'variables' and ten 'cases', with one or two zeros in a row and each row sum equal to four. Apply the `mix.ZeroReplaceSimp` command, first with imputed value equal to 1, second with imputed values vector `de=(1:10)*.1` i.e. comprising values 0.1, 0.2, . . . , and 1.

```
$tit
[1] ''One, two zeros in a row and each row sum is 4.''

$sum
[1] 4

$sta
[1] 0
```

```
$mat
   aa bb cc dd ee
1   1  1  1  1  0
2   1  1  1  0  1
3   1  1  0  1  1
4   1  0  1  1  1
5   0  1  1  1  1
6   0  0  2  1  1
7   1  0  0  2  1
8   1  1  0  0  2
9   2  1  1  0  0
10  0  2  1  1  0

$class
[1] ``mixture''

> mix.ZeroReplaceSimp(m,1)

$tit
[1] ``One, two zeros in a row and each row sum is 4.''

$sum
[1] 4

$sta
[1] 2

$mat
       aa        bb        cc        dd        ee
1  0.8000000 0.8000000 0.8000000 0.8000000 0.8000000
2  0.8000000 0.8000000 0.8000000 0.8000000 0.8000000
3  0.8000000 0.8000000 0.8000000 0.8000000 0.8000000
4  0.8000000 0.8000000 0.8000000 0.8000000 0.8000000
5  0.8000000 0.8000000 0.8000000 0.8000000 0.8000000
6  0.6666667 0.6666667 1.3333333 0.6666667 0.6666667
7  0.6666667 0.6666667 0.6666667 1.3333333 0.6666667
8  0.6666667 0.6666667 0.6666667 0.6666667 1.3333333
9  1.3333333 0.6666667 0.6666667 0.6666667 0.6666667
10 0.6666667 1.3333333 0.6666667 0.6666667 0.6666667

$class
[1] ``mixture''

> round(mix.ZeroReplaceSimp(m, (1:10)*.1,
    Col=FALSE)$mat, 2)
    aa   bb   cc   dd   ee
1  0.98 0.98 0.98 0.98 0.10
2  0.95 0.95 0.95 0.19 0.95
3  0.93 0.93 0.28 0.93 0.93
4  0.91 0.36 0.91 0.91 0.91
5  0.44 0.89 0.89 0.89 0.89
6  0.46 0.46 1.54 0.77 0.77
7  0.74 0.52 0.52 1.48 0.74
8  0.71 0.71 0.57 0.57 1.43
9  1.38 0.69 0.69 0.62 0.62
10 0.67 1.33 0.67 0.67 0.67
```

In the second command the simple replacement strategy is used, inputing values 0.1, 0.2 ... and 1 in rows from 1 to 10 and the results are rounded to two decimals. It should be noted that the last line is the same (and it should be) as in the former case, where the imputed value was 1 in each column.

The routine for Multiplicative replacement strategy – see Martin-Fernandez *et al.* (2003) `mix.ZeroReplaceMult(m, de, eps=1e-6)` parameters are

`m`	the mix object,
`de`	vector of imputed values, if the length of `de` is smaller than the number of columns i.e. variables, the `de` is repeated to adequate length, and
`eps`	value for the 'zero' checking.

Output is a mix object where each 'zero' value is replaced and the row sums are preserved.

Example

The `mix.ZeroReplaceMult` is applied to the mix object `m` from the previous example.

```
> mix.ZeroReplaceMult(m,1)

$tit

[1] "One, two zeros in a row and each row sum is 4."

$sum
[1] 4

$sta
[1] 2

$mat
     aa   bb   cc   dd   ee
1  0.75 0.75 0.75 0.75 1.00
2  0.75 0.75 0.75 1.00 0.75
3  0.75 0.75 1.00 0.75 0.75
4  0.75 1.00 0.75 0.75 0.75
5  1.00 0.75 0.75 0.75 0.75
6  1.00 1.00 1.00 0.50 0.50
7  0.50 1.00 1.00 1.00 0.50
8  0.50 0.50 1.00 1.00 1.00
9  1.00 0.50 0.50 1.00 1.00
10 1.00 1.00 0.50 0.50 1.00

$class
[1] ``mixture''
```

With the Multiplicative replacement strategy the value of the data replacing the zeros is the exact imputed value, i.e. 1.

Example

In the glacial dataset there are 48 zero values in the Crystalline and Miscellaneous variables. Because this kind of zero is usually understood as 'a trace too small to measure', it seems reasonable to replace them by a suitably small value. The smallest value detected is used to replace the zero value.

The command `mix.Sub(m,5)` produces the four-part normalized subcomposition and the `min` command of the third and of the fourth column produces the lowest detected value equal to 0.001. Then the Multiplicative replacement strategy is applied.

```
> mix.CheckData(m, Detect=T)
zero values in the data
zero value column 3 row 4  value 0 No. 1
zero value column 3 row 14 value 0 No. 2
```

```
... .... ....
zero value column 4 row 90 value 0 No. 47
zero value column 4 row 92 value 0 No. 48
No negative values in the data
$tit
[1] ``Noting -1 for neg., 0 for zero and 1
      for positive values''
$sum
[1] NA
$sta
[1] 0
$mat
       A  B  C  D Count
[1,]   1  1  1  0     1
[2,]   1  1  1  1     1
[3,]   1  1  1  1     1
[4,]   1  1  0  1     1
 ...       ...       ...
[90,]  1  1  1  0     1
[91,]  1  1  1  1     1
[92,]  1  1  1  0     1
$class
[1] ``mixture''
> m5 <- mix.Sub(m,5)
> mm <- mix.ZeroReplaceMult(m5,.001)
$tit
[1] ``GLACIAL DATA 92 samples of pebbles
      of glacial tills sorted into four
      categories percentages by weight''
$sum
[1] 1
$sta
[1] 3
$mat
           A           B           C           D
1  0.91708200  0.070929000  0.010989000  0.001000000
2  0.88900000  0.101000000  0.005000000  0.005000000
3  0.87300000  0.109000000  0.008000000  0.010000000
4  0.84515400  0.134865000  0.001000000  0.018981000
... ... ...
90 0.15884100  0.832167000  0.007992000  0.001000000
91 0.17192269  0.755849440  0.012207528  0.060020346
92 0.31368600  0.658341000  0.026973000  0.001000000
attr(,``class'')
[1] ``mixture''
```

The same result can be obtained using a slightly more complicated procedure:

```
> m <- mix.Sub(m, 5, Normalize=F)
> m <- mix.ZeroReplaceMult(m, .1)
> m <- mix.Normalize(m)
```

It must be emphasized that the straightforward

```
> mix.ZeroReplaceMult(mix.Sub(m,5),.1)
```

is inadequate because the `mix.Sub` routine normalizes the data and the 0.1 imputed value is not adequate any more. The values in the rows where there were zero values (first, fourth ...) will now show new proportions, not at all coherent with the original data.

Example

From the glacial data mix object `mm` a mix object will be constructed with the pebbles counts for the cases names. Again a dendrogram and tetrahedron will be drawn with the data points, where classes will be marked with different colours (see Figs 8 and 9).

```
> dimnames(mm$mat)[[1]]<- m$mat[,5]
> mm
$tit
[1] ``b.glacial.dat''
$sum
[1] 1
$sta
[1] 0
$mat
           A           B           C           D
282 0.91708200 0.070929000 0.010989000 0.001000000
368 0.88900000 0.101000000 0.005000000 0.005000000
607 0.87300000 0.109000000 0.008000000 0.010000000
532 0.84515400 0.134865000 0.001000000 0.018981000
... ...        ...         ...         ...
245 0.15884100 0.832167000 0.007992000 0.001000000
575 0.17192269 0.755849440 0.012207528 0.060020346
698 0.31368600 0.658341000 0.026973000 0.001000000
$class
[1] ``mixture''
> d <-mix.Ait(mm)
> hc<-hclust(d, method=``complete'',
    members= NULL)
> plot(hc, labels =m$mat[,5], hang=0.1, main=
    ``Cluster Dendrogram'', ylab=``Aitchison
    distance'')
> mix.Quad2kin(`glac5.kin',mm,
    clu=cutree (hc,5), scale=0.14)
> mix.Quad2kin(`glac8.kin',mm,
    clu=cutree (hc,8),scale=0.14)
```

Conclusions

Some MixR routines for reading different data file formats have been presented, together with an explanation of visualization of three- and four-parts

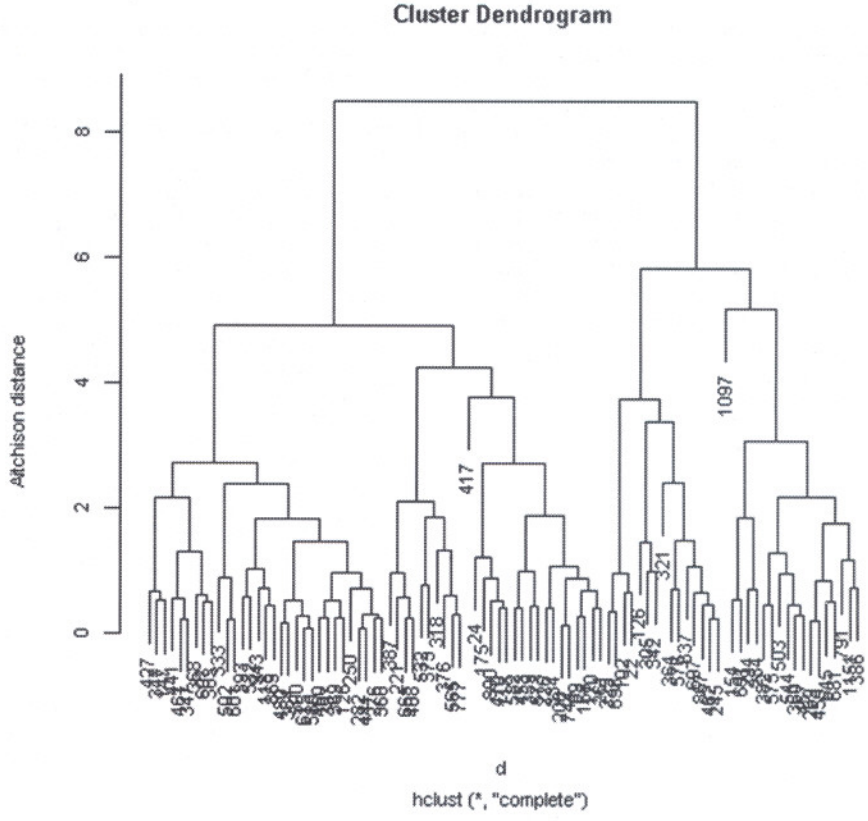

Fig. 8. Glacial dataset classification represented by a dendrogram.

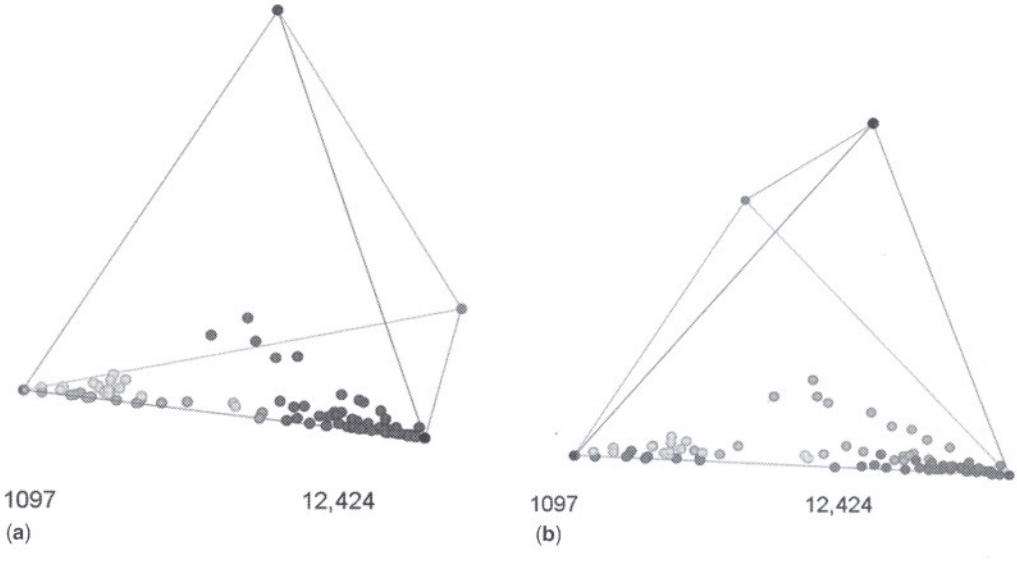

Fig. 9. Two snapshots of glacial data 3D display in the tetrahedron. The snapshot of 3D KiNG view of the glacial data is classified into (**a**) five classes and (**b**) eight classes.

(sub)compositions with additional features and also the computation of distances between compositional data. The use of some R routines for classification and plotting dendrograms has also been demonstrated. The mix routines are available at http://vlado.fmf.uni-lj.si/pub/MixeR

With the authors of the 'compositions' package, support is provided for a complementary use of 'compositions' package and MixeR routines – transformations from the mix object to the objects of the four different classes: rplus, rcomp, acomp and aplus are implemented in the 'compositions' package and the transformations from the four classes to the mix objects can be found in MixeR library routines. With these routines it is hoped to enable users to apply and to benefit from both, the 'compositions' package and also the MixeR library routines.

References

AITCHISON, J. 1986. *The Statistical Analysis of Compositional Data.* Chapman & Hall, New York.

VAN DER BOOGAART, K. G. & TOLOSANA, R. 2005. The compositions Package. World Wide Web address: http://cran.r-project.org/src/contrib/Descriptions/compositions.html.

VAN DER BOOGAART, K. G. & TOLOSANA-DELGADO, R. 2006. Compositional data analysis with 'R' and the package 'compositions'. *In*: BUCCIANTI, A., MATEUV-FIGUERAS, G. & PAWLOWSKY-GLAHN, V. (eds) *Compositional Date Analysis in the Geosciences: From theory to practice*. Geological Society, London, Special Publications, **264**, 119–127.

BREN, M. & BATAGELJ, V. 2003. Compositional data analysis with R. *In*: *Proceedings CoDaWork'03, Girona, Spain*, 111–122.

MARTÍN-FERNÁNDEZ, J. A., BARCELO-VIDAL, C., BREN, M. & PAWLOWSKY-GLAHN, V. 1999. A measure of difference for compositional data based on measures of divergence. *In*: *Proceedings of the 5th Annual Conference of the International Association for Mathematical Geology, Trondheim, Norway*, **1**, 211–215.

MARTÍN-FERNÁNDEZ, J. A., BARCELO-VIDAL, C. & PAWLOWSKY-GLAHN, V. 2003. Dealing with zeros and missing values in compositional data sets using nonparametric imputation. *Mathematical Geology*, **35** (3), 253–278.

REYNOLDS, J. H. & BILLHEIMER, D. 2002. Basic Compositional Data Analysis Functions for S+/R. World Wide Web Address: http://www.biostat.wustl.edu/archives/html/s-news/2003-12/msg00139.html.

THIÓ-HENESTROSA, S. & MARTÍN-FERNÁNDEZ, J. A. 2006. Detailed guide to CoDaPack: a freeware compositional software. *In*: BUCCIATI, A., MATEUV-FIGUERAS, G. & PAWLOWSKY-GLAHN, V. (eds) *Compositional Data Analysis in the Geosciences: From theory to practice*. Geological Society, London, Special Publication, **264**, 101–118.