Compositional data analysis with 'R' and the package 'compositions'

K. G. VAN DER BOOGAART[1] & R. TOLOSANA-DELGADO[2]

[1]*Institut für Mathematik und Informatik, Ernst-Moritz-Arndt-Universität Greifswald, Greifswald D-17487, Germany (e-mail: boogaart@uni-greifswald.de)*

[2]*Departament Informàtica i Matemàtica Aplicada, Universitat de Girona, Girona E-17071, Spain*

Abstract: This paper is a hands-on introduction and shows how to perform basic tasks in the analysis of compositional data following Aitchison's philosophy, within the statistical package 'R' and using a contributed package (called 'compositions'), which is devoted specially to compositional data analysis. The studied tasks are: descriptive statistics and plots (ternary diagrams, boxplots), principal component analysis (using biplots), cluster analysis with Aitchison distance, analysis of variance (ANOVA) of a dependent composition, some transformations and operations between compositions in the simplex.

This paper will show how the basic tasks of compositional data analysis (Aitchison *et al.* 2002) can be performed with the package 'compositions' in the free statistical environment 'R' (R Development Core Team 2003). The paper aims to be useful for a wide spectrum of 'R' users: for this reason, it is suggested that the experienced skip these first steps, whereas those who never heard about 'R' should begin with Appendix A before continuing with the text. It is strongly recommended that the reader be in front of the computer, typing the examples outlined here: thus, text output of these instructions is kept to a minimum, and almost all figures are not included, although they are described briefly (with a few exceptions).

First steps

'R' is a powerful computer environment for multipurpose statistics and data analysis. It is available for all computer platforms and can be downloaded from 'http://www.cran.R-project.org'. 'Compositions' is a contributed package for 'R', devoted specially to the analysis of compositional data; it can be downloaded from 'http://www.stat.boogaart.de/compositions'. 'R' and 'compositions' are both distributed and developed under the GNU public license, hence they are available free of charge. Further instructions on downloading, installation and getting started with the software can be found in Appendix A.

'R' is classically based on a command line interface, but various graphical user interfaces are available from 'http://www.cran.R-project.org'. When compared with other compositional software, the 'R' package provides a maximum of flexibility. However, being based on a computer language, it demands from its users not to be afraid of reading manuals or of typing to a command line any command found out there. However, it should be remembered that 'R' and its packages are a living project permanently adapted to the development of the field. More intstructions can be found at 'http://www.stat.boogaart.de/compositions/'.

After starting 'R' (either by clicking on the appropriate icon, selecting the entry 'R' in the start menu or by typing the command 'R' to a console or command window, after installing the software) a command window appears where commands can be given to 'R'. The following appears:

```
R: Copyright 2004, The R Foundation for
Statistical Computing Version 2.0.1
(2004-11-15), ISBN 3-900051-07-0

R is free software and comes with
ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it
under certain conditions.

Type 'license()' or 'licence()' for
distribution details.

R is a collaborative project with
many contributors.

Type 'contributors()' for more
information and 'citation()' on how
to cite R or R packages in
publications.

Type 'demo()' for some demos,
'help()' for on-line help, or 'help.
start()' for a HTML browser
interface to help.
Type 'q()' to quit R.

>
```

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**, 119–127.
0305-8719/06/$15.00

The version number should be checked, since at least version 2.0.0 is required for running compositions. The '>' mark shows that 'R' is willing to accept commands. This character should not be typed with the commands. To see how 'R' works, type '3*7', and hit the ENTER-Key to make 'R' execute this command:

```
> 3*7
[1] 21
>
```

'R' executes the command by multiplying 3 and 7 and then prints the result 21. At this moment ignore the '[1]'. 'R' can in this way be used as a (extremely powerful) calculator. To prepare 'R' for compositional data analysis the library compositions must be loaded with the library command:

```
> library(compositions)
Attaching package 'compositions':
  The following object(s) are masked
    from package:stats:
    cor cov dist var
  The following object(s) are masked
    from package:base:
    %*%
>
```

Either such output, or no output at all, informs the user about a properly loaded package. When an error appear such as this:

```
> library(compositions)
Error in library(compositions):
  There is no package called
  'compositions'
```

this means that the package is not properly downloaded or installed. Instructions for downloading and installation of the package can be found in Appendix A.

After loading the package, some example data from the package should be loaded with the 'data' command:

```
> data(SimulatedAmounts) # Load
    example data (no output)
> ? SimulatedAmounts # Show help
    about example data
```

Note that a hash mark '#' denotes the beginning of a comment: after it, the rest of the line is ignored by 'R'. Therefore, it is not necessary to type them.

When working in a terminal, the help can be closed by typing 'q' for *Q*uit. In a windows-based environment the help window can simply be closed.

```
> ls()# Show names of all variables/
    datasets
[1] "sa.dirichlet" "sa.dirichlet.
      dil" "sa.dirichlet.mix"
[4] "sa.dirichlet5" "sa.dirichlet5.
      dil" "sa.dirichlet5.mix"
... (lines omitted)
```

The other commands show a typical usage of 'R': Use ? to get help information, or 'ls()' to show all variables/datasets defined previously. Just type the name of a dataset to show its content, which in this case is a set of simulated amounts of three different chemical elements in ppm:

```
> sa.lognormals # Show one of the
    datasets
            Cu          Zn          Pb
[1,] 8.8043262 35.1671810 45.895025
[2,] 0.8115227  2.6547329 47.804310
[3,] 1.2836130 12.4472047 40.553628
... (lines omitted)
[60,] 3.9854998 6.1301909 40.579417
```

To edit or just to inspect the dataset in a spreadsheet-like environment the command 'fix(sa.lognormals)' may be used. Appendix A contains instructions on how to load datasets.

Basic compositional data analysis

The **zero step** when using the package is to mark your data explicitly as a set of elements from a simplex under Aitchison geometry (Aitchison *et al.* 2002). This is done by converting the dataset to an Aitchison compositional set through the function 'acomp', and storing it into a new variable by using the assignment sign '<-'.

```
> data(SimulatedAmounts) # just in
    case you start here
> cdata <- acomp(sa.lognormals)
> cdata
              Cu          Zn          Pb
[1,] 0.097971136 0.391326782 0.51070208
[2,] 0.015828238 0.051778890 0.93239287
[3,] 0.023646054 0.229295970 0.74705798
... (lines omitted)
[60,] 0.078617049 0.120922730 0.80046022
attr(,"class")
[1] "acomp"
```

The dataset is now closed. Note that the closure constant is automatically considered as one. In this case, the resulting object is stored in 'cdata' and marked as an Aitchison composition (having the attribute class 'acomp') such that it is automatically treated in an adequate way in further commands. For example, the 'plot' command will automatically draw a ternary diagram:

```
> plot(cdata)  # Ternary diagram
> ? plot.acomp # Help on the acomp -
    specific plot function
```

One quits the help by closing the help window or by typing q for 'quit'.[1] There is always an example of the command at the end of each help page. Try this out to see what happens.

Another graphical display of compositional data related closely to the Aitchison geometry of the simplex and displaying the Aitchison distance in a visual way is the boxplot of log-ratios:

```
> boxplot(cdata) # Box-plots of
    pairwise ratios in log scale
> ? boxplot.acomp # Help on
    compositional box-plots
> boxplot(cdata,log=FALSE) # use
    normal scale
```

As a result a square table of boxplots appears, displaying the ratios of the row and the column parts in log scale. Various types of descriptive statistics can be computed by intuitive commands:

```
> mean(cdata)
        Cu          Zn          Pb
0.08918175 0.23949922 0.67131903
attr(,"class")
[1] "acomp"
```

The result is the mean in Aitchison geometry (i.e. closed geometric mean), which is again a composition. This single composition can be displayed by a pie-chart or a barplot:

```
> pie(mean(cdata))
> barplot(mean(cdata))
```

[1]For some commands (barplot, boxplot, cdt, cor, cov, idt, mean, names, perturbe, plot, power, princomp, qqnorm, rnorm, runif, scale, segments, split, summary, var, +, −, *, /, %*% you need to add '.acomp' to see the Aitchison compositional specific help.

A barplot can also be used to display the whole dataset:

```
> barplot(cdata) # Display the whole
    data set
```

The variation of compositions can be summarized in several ways (Aitchison *et al.* 2002; Pawlowsky-Glahn & Egozcue 2001):

```
> variation(cdata) # Variation matrix
           Cu         Zn         Pb
Cu 0.0000000 0.4046994 2.938182
Zn 0.4046994 0.0000000 2.910539
Pb 2.9381816 2.9105389 0.000000
> var(cdata) # Variance matrix of the
    clr-transform
           Cu          Zn          Pb
Cu  0.4194692  0.2125124 -0.6319817
Zn  0.2125124  0.4102550 -0.6227674
Pb -0.6319817 -0.6227674  1.2547491
> mvar(cdata)     # metric variance
[1] 2.084473
>msd(cdata)       #  metric  standard
  deviation = sqrt(mvar/(D-1))
[1] 1.0209
> summary(cdata) # multiple
    information about
    pairwise ratios
...
```

Indented lines are continuations of the previous line.

A graphical way to display the variability of a composition is the biplot, based on principal

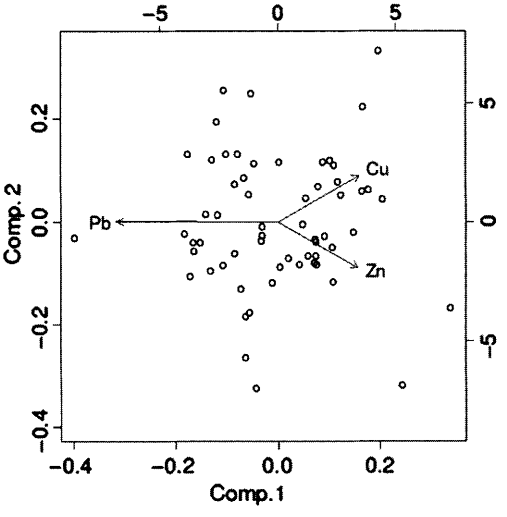

Fig. 1. Biplot of a three-part composition (Cu, Zn, Pb).

component analysis, which uses the clr transforms (Aitchison 2002):

```
> pca <- princomp(cdata) # perform
    PCA and store the result in pca
> pca       # display results as text
Call:
princomp.acomp(x = cdata)

Standard deviations:
   Comp.1      Comp.2
1.3604382 0.4460269

3 variables and 60 observations.
Mean (compositional):
        Cu          Zn          Pb
0.08918175 0.23949922 0.67131903
attr(,"class")
[1] "acomp"
+Loadings (compositional):
             Cu        Zn        Pb
Comp.1 0.5533583 0.5570883 1.8895534
Comp.2 0.4207858 1.7307697 0.8484445
attr(,"class")
[1] "acomp"
-Loadings (compositional):
             Cu        Zn        Pb
Comp.1 1.312246 1.3034604 0.3842932
Comp.2 1.725060 0.4193976 0.8555428
attr(,"class")
[1] "acomp"
> screeplot(pca) # display
    importance of components
> biplot(pca) # display direction of
    components
```

The last component always has no importance. The first component, giving the highest variation, corresponds to the Pb against Zn and Cu balance here (as can be seen in Fig. 1). It explains 90% of the variability, as can be obtained by considering the variance of first component divided by the metric variance of 'cdata'.

Working with compositions of four or more parts

To analyse a different dataset or a subcomposition you might assign something different to 'cdata' or any other variable representing your compositional dataset, e.g. by

```
> data(SimulatedAmounts) # Load the
    example datasets
> sa.groups5 # One of these
> cdata <- acomp(sa.groups5,parts=c
    ("Cd", "Pb", "Co","Cu"))
> cdata
```

The optional parameter 'parts=', allows you to select the parts to be used in the subcomposition. Optional parameters are a typical way of 'R' providing additional functionality to the default behaviour of a command. The possible optional parameters and their effects are documented in the help to each command that can be invoked by '? *nameofunction*'. The 'c()' function is just here to Concatenate the variable names.

In principle, now everyone of the aforementioned commands can be applied to the new dataset. Try:

```
> plot(cdata)
```

Since a ternary diagram can display only three parts at the same time, a table of multiple ternary diagrams, containing subcompositions or marginal compositions (Fig. 2) must be displayed. As a default, two parts are determined by the row and the column occupied by each plot, and the geometric mean of the remaining components is taken as the third component. Alternatively one can specify a component, by using the optional parameter 'margin':

```
plot(cdata,margin="Cd")
```

Performing a cluster analysis with Aitchison distance

A hierarchical cluster analysis can be performed with the following instructions. First the clustering must be computed and the result stored in a variable:

```
> Clusters <- hclust(dist(cdata),
    method="complete") # compute
    clustering
```

Since 'cdata' is marked as an Aitchison composition, 'dist' automatically computes the Aitchison distance. The linkage method of clustering (here 'complete') can be replaced by any other method (e.g. 'single') as described in the help '? hclust'. Now the results should be displayed (Fig. 3):

```
plot(Clusters)  # shows the dendrogram
```

When the user has decided on the number of groups to interpret, maybe four in this case, a new variable containing the groups assigned to each case can be generated and the group membership

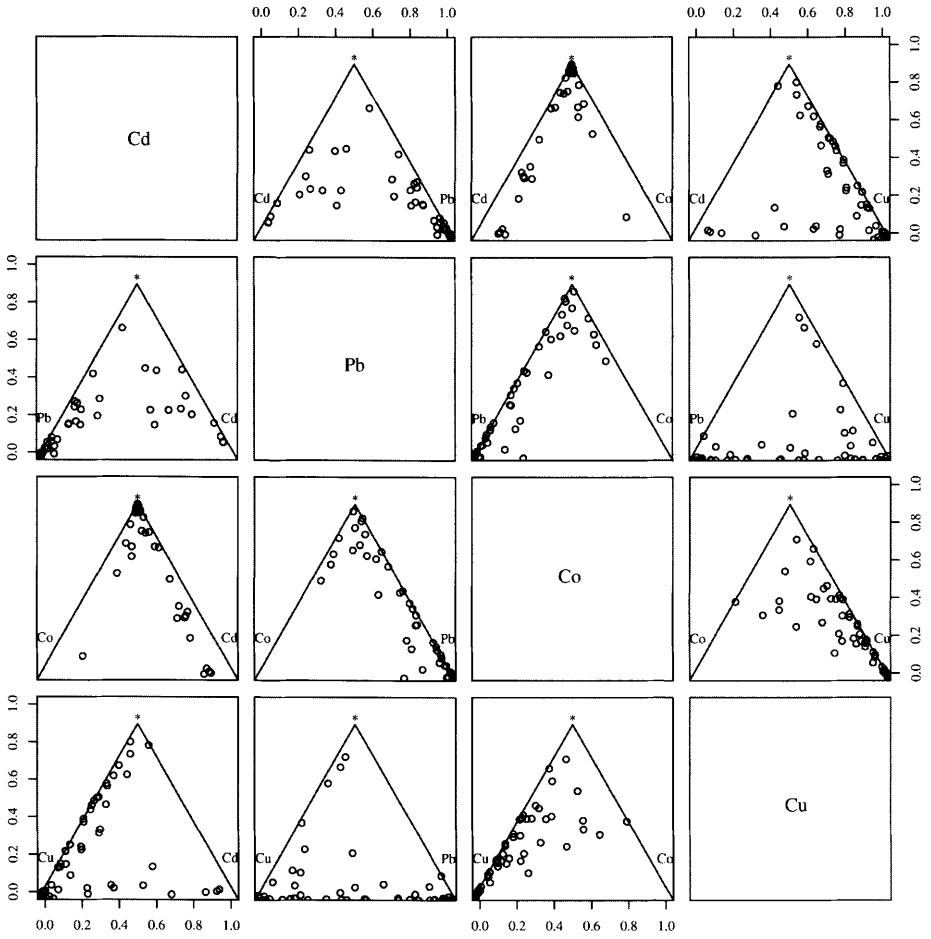

Fig. 2. Matrix of ternary diagrams of a four-part composition (Cd, Pb, Co, Cu).

can be displayed in ternary diagrams, boxplots and biplots.

```
> group <- cutree(Clusters,4)
> group
[1] 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 1 1 2
  2 1 3 2 3 2 4 3 3 2 2 4 3 4 2 2 3 2
  2 2
[39] 3 1 3 3 2 4 4 3 4 4 4 3 4 4 3 4 4
  4 3 3 4 4
> plot(cdata,col=group)
> plot(cdata,pch=group)
> plot(cdata, pch=as.character
    (group))
> plot(cdata,col=group,center=T)
    #display centered data
> plot(Clusters,labels=group)
> biplot(princomp(cdata), xlabs=
    group)
> boxplot(cdata,factor(group))
```

Compositional computation

Various mathematical transformations and operations are defined for the Aitchison simplex (Aitchison *et al.* 2002). These are interesting mainly for developers of new statistical methods. The perturbation and the power transform are considered as addition and scalar multiplication in a vector space structure of the simplex,

```
> a <- acomp(c(1,2,2)) # a single
    composition
> a
[1] 0.2 0.4 0.4
attr(,"class")
[1] "acomp"
> b <- acomp(c(8,1,1)) # a second
    composition
```

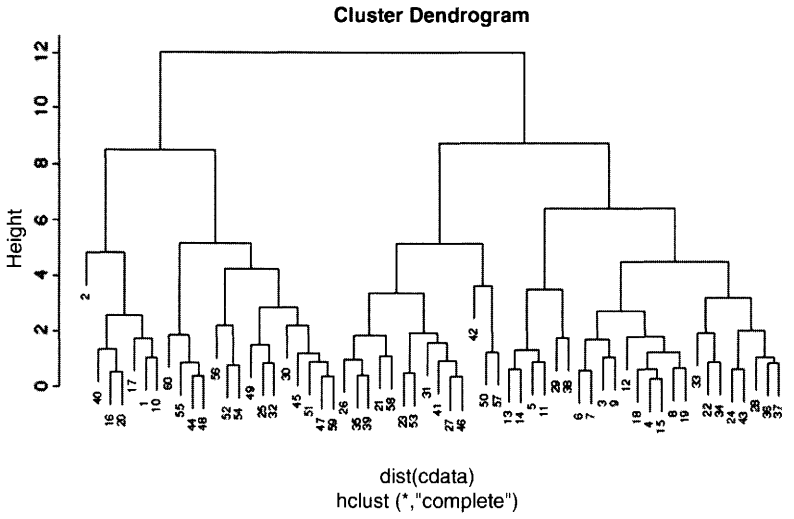

Fig. 3. Dendrogram of groups in a four-part composition (Cd, Pb, Co, Cu), defined by the Aitchison distance.

```
> b
[1] 0.8 0.1 0.1
attr(,"class")
[1] "acomp"
> a+b # adding is perturbation
[1] 0.6666667 0.1666667 0.1666667
attr(,"class")
[1] "acomp"
> 2*a # multiplication is power
    transform
[1] 0.1111111 0.4444444 0.4444444
attr(,"class")
[1] "acomp"
> (a+a)/2-a # inverse operations
[1] 0.3333333 0.3333333 0.3333333
attr(,"class")
[1] "acomp"
> xx<- (cdata-mean(cdata))/msd(cdata)
    # parallel on the whole data set
> xx
...
> mean(xx)            # xx is centered
  Cd   Pb   Co   Cu
0.25 0.25 0.25 0.25
attr(,"class")
[1] "acomp"
> msd(xx)             # and normalized
[1] 1
> a %*% a             # scalar product
[1] 0.320302
> norm(a)             # norm
[1] 0.5659523
> cdata %*% acomp(c(1,2,3,4))
    # scalar products
[1]0.43526714 -1.53700304 2.40682769
   1.61307672  2.64960822 2.38350333
... (lines omitted)
> yy <- chol(solve(clrvar2ilr (var
    (cdata)))) %*% cdata # matrix
    operation
```

```
> var(yy) # var^-0.5 * cdata
      [,1]   [,2]  [,3]  [,4]
[1,]  0.75 -0.25 -0.25 -0.25
[2,] -0.25  0.75 -0.25 -0.25
[3,] -0.25 -0.25  0.75 -0.25
[4,] -0.25 -0.25 -0.25  0.75
```

The standard transforms can be computed by

```
> CenteredLogRatio  <- clr(cdata)
> IsometricLogRatio <- ilr(cdata)
> AdditiveLogRatio  <- alr(cdata)
> OriginalData      <- clr.inv(CenteredLogRatio)
> OriginalData      <- ilr.inv(IsometricLogRatio)
> OriginalData      <- alr.inv(AdditiveLogRatio)
> CenteredLogRatio
            Cd          Pb          Co          Cu
[1,] -1.62815906  2.82253578 -0.46785026 -0.72652645
[2,] -0.28013715  2.27945644  1.20048072 -3.19980002
... lines omitted
[60,] 1.72468264 -2.56743127 -0.16183132 1.00457995
attr(,'class')
[1] 'rmult'
```

Working with grouped data

Sometimes a group membership is available. In the case of the 'sa.groups5' dataset, it represents a sector of an imaginary river where the sample was originally collected. This information is stored in the variable 'sa.groups5.area':

```
plot(acomp(sa.groups5),
  col=sa.groups5.area)
boxplot(acomp(sa.groups5),
  sa.groups5.area)
```

When the user wants to analyse only one of the groups, a subset of the data is selected based on a criterion:

```
> sa.groups5.area
 [1] Upper Upper Upper Upper Upper
       Upper Upper Upper Upper Upper
[11] Upper Upper Upper Upper Upper
       Upper Upper Upper Upper Upper
[21] Middle Middle Middle Middle
       Middle Middle Middle Middle
       Middle Middle
[31] Middle Middle Middle Middle
       Middle Middle Middle Middle
       Middle Middle
[41] Lower Lower Lower Lower Lower
       Lower Lower Lower Lower Lower
[51] Lower Lower Lower Lower Lower
       Lower Lower Lower Lower Lower
Levels: Lower Middle Upper
> upper <- split(cdata,sa.groups5.
    area) [["Upper"]]
> plot(upper)
> mean(upper)
```

A parallel analysis of all groups is possible through the 'lapply' or 'sapply'-function of 'R':

```
> sapply(split(cdata,sa.groups5.area),
    mean)
         Lower       Middle        Upper
Cd 0.064366655 0.006801803 0.001636658
Pb 0.059178242 0.572889919 0.957191909
Co 0.009509995 0.002658064 0.004323912
Cu 0.866945108 0.417650214 0.036847521
```

However the grouping information could be used to check whether the groups are really different in a Multivariate Analysis of Variance (manova), which can be done by 'R' standard routines based on the ilr transform:

```
> m <- manova(ilr(cdata)~sa.groups5. area)
> summary(m)
                Df Pillai approx F num Df   den
   Df Pr(>F)
sa.groups5.area 2 1.0872 22.2312 6
   112 < 2.2 e-16 ***
Residuals 57
---
Signif. codes: 0 "***" 0.001 "**" 0.01 "*"
   0.05"." 0.1 " "1
> plot(ilr.inv(residuals(m)),col=sa.groups5.
    area)
> plot(ilr.inv(predict(m)),col=sa.groups5.
    area)
> qqnorm(ilr.inv(residuals(m)))
> mvar(predict(m))/(mvar(residuals(m)
   +predict (m))) # "R"^2
[1] 0.3980416
> diag(ilrvar2clr(var(predict(m)))/ilrvar2
   clr(var(residuals(m)+predict(m))))
[1] 0.4001846 0.5670027 0.1392320 0.2654141
```

Here one sees a highly significant influence of the group given by a p-value stated as '<2.2e-16'. If this example was run, a series of plots would result: the first one would show the residuals with substantial spread. The second plot shows the location of the predicted group means in ternary diagrams. Unfortunately, the variable names are lost during the ilr transform, such that the plots are drawn without labels. The third plot shows qqnorm-plots of the pairwise log-ratios, in order to check the normality assumption used in the manova. The last two calculations give the total of the model of about 39% and the individual's for the four parts of the composition.

In a similar way a discrimination analysis can be performed based on the ilr transform and standard functionality of 'R':

```
> library(MASS) # Loading
    appropriate library
> # Generating example data
> subsample <- sample(1:60,45)
> TrainingData
      <- acomp(cdata[subsample,])
> TrainingGroups
      <- sa.groups5.area[subsample]
> ControlData
      <- acomp(cdata[-subsample,])
> ControlGroups
       <- sa.groups5.area[-subsample]
> ControlGroups
[1] Upper Upper Upper Middle Middle
      Middle Middle Middle Middle
      Middle
[11] Lower Lower Lower Lower Lower
       Levels: Lower Middle Upper
> # Performing the discriminat
    analysis
> dscr <- lda(TrainingGroups~.,ilr
    (TrainingData)) # Discrimination
    Analysis
> dscr
... (output omitted)
> predict(dscr,newdata=ilr
    (ControlData)) # Classify
    ControlData
$class
[1] Upper Upper Upper Middle Middle
      Lower Middle Middle Middle
      Middle
[11] Lower Lower Lower Lower Lower
       Levels: Lower Middle Upper
$posterior
        Lower         Middle      Upper
1 3.626286e-16 1.851031e-07 9.999998e-01
2 7.991869e-12 8.473827e-05 9.999153e-01
... (lines omitted)
> table(ControlGroups, predict
    (dscr, newdata=ilr (ControlData))
    $class)
ControlGroups Lower Middle Upper
       Lower  5      0       0
       Middle 1      6       0
       Upper  0      0       3
```

The calculated classification of the 15 control samples based on 45 training samples was thus correct, with one exception. More detailed information about discriminant analysis and 'lda' function can be found in the 'R' help.

Conclusions

For the beginner, this approach immediately provides all basic compositional plots, summaries and transformation in the form of simple standard commands given in this publication. More helpful reading can be found in *Using the R package 'compositions'*, available at http://www.stat.boogaart.de/compositions. Users can perform advanced analysis using the package in combination with the statistical sub-routines of 'R' as exemplified in the later chapters and experts can even extend the functionality through the programming interface of 'R'. The authors are open to suggestions to include more functionality and convenience to the package.

Appendix A: Help with technical details

Downloading and installing 'R'

On 'http://www.cran.R-project.org' one can find detailed instruction on downloading and installing 'R', as well as the downloadable packages themselves. Users must download the setup program of the base part of a precompiled binary distribution of 'R' for their platform.

For example, for windows users it is sufficient to download the 'rw? ? ? ? .exe' file from 'http://www.cran.R-project.org/bin/windows/base/' and to double click the downloaded file to start the installation process.

Downloading and installing 'compositions'

The package is available from 'http://www.stat.boogaart.de/compositions/' or as a contributed package from 'http://www.cran.R-project.org'. In a windows system ('R' v.2.0.1 or later) the package can now be installed through the 'Packages' menu option 'Install package(s) from local zip file...'. On Unix/Linux systems it is done by the command:

```
"R" CMD INSTALL
  DownloadedPackage.tar.gz
```

with 'DownloadedPackage.tgz' replaced by the actual filename of the downloaded package.

Importing data to 'R'

The most simple way to provide data to 'R' is to store them into a simple text file. The first row should contain the variable names seperated by a semicolon. The following lines contain the data, again separated by a semicolon.

```
Cd;Zn;Pb;Cd;Co
1.2;2.6;4.9;0.2;5
23.4;11;0.2;0.002;6.2
...
```

The data can then be loaded by the 'R'-commands:

```
> mydata <- read.csv("C:/
    mydirectory/myfile.txt",
    sep=";", dec=".")
> fix(mydata) # you must close
    the window afterwards
```

One should always check with the fix command that the data are properly loaded before using them. Directories must always be separated by a forward slash '/' in pathnames. All spreadsheet programs can export to this format, when instructed to store as '.csv'. The separator and the decimal symbol can vary with the local configuration of the computer. Note that 'R' only uses a dot as the floating comma symbol in any output, although the import procedure accepts the optional parameter 'dec=',' to deal with the colon (see '? read.csv' or '? read.table'). To use Tabulator as a separating character use 'sep='' in the 'read.csv' command.

Newbee problems and solutions

- 'R' is not exactly made for beginners: it's not your fault. Try to find someone to help you. Next year you will be the expert.
- When 'R' does not find your file, give the whole path and separate the directories with '/', not with a backslash. Don't forget the extension (e.g. '.txt').
- Wear glasses when copying and typing commands.
- When you get neither a plot nor an error, the plot window is probably iconified.
- When 'R' answers with '+' instead of '>', you have made a typing error and 'R' thinks that the command is not yet finished. Type ';', the ENTER-key, and try again.
- When you are bored by retyping commands again and again, try the up and down arrow keys or copy and paste from your favourite editor or a script window.

- It doesn't work in the second session: Have you loaded all necessary libraries and prepared all variables?
- 'R' comes with plenty of help. Type the `'help.start()'` command and start with 'Introduction to R'.
- Type `'q()'` for quit and the ENTER-key to leave 'R'. Save your workspace, when asked.

References

AITCHISON, J. 2002. Simplicial inference. *In*: VIANA, M. A. G. & RICHARDS, D. S. P. (eds) *Algebraic Methods in Statistics and Probability*. Contemporary Mathematics Series, **287**, American Mathematical Society, Providence, Rhode Island, 1–22.

AITCHISON, J., BARCELÓ-VIDAL, C., EGOZCUE, J. J. & PAWLOWSKY-GLAHN, V. 2002. A concise guide to the algebraic geometric structure of the simplex, the sample space for compositional data analysis. *In*: BAYER, U., BURGER, H. & SKALA, W. (eds) *Proceedings of the 8th Annual Conference of the International Association for Mathematical Geology*, Berlin, Germany, 387–392.

PAWLOWSKY-GLAHN, V. & EGOZCUE, J. J. 2001. Geometric approach to statistical analysis on the simplex. *Stochastic Environmental Research and Risk Assessment*, **15** (5), 384–398.

R Development Core Team 2003. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria (http://www.R-project.org).