Use R!

K. Gerald van den Boogaart
Raimon Tolosana-Delgado

Analyzing Compositional Data with R

Springer

Use R!

Series Editors
Robert Gentleman Kurt Hornik Giovanni G. Parmigiani

For further volumes:
http://www.springer.com/series/6991

K. Gerald van den Boogaart
Raimon Tolosana-Delgado

Analyzing Compositional Data with R

Springer

K. Gerald van den Boogaart
Raimon Tolosana-Delgado
Freiberg for Resources Technology
Helmholtz Institute
Freiberg
Germany

ISBN 978-3-642-36808-0 ISBN 978-3-642-36809-7 (eBook)
DOI 10.1007/978-3-642-36809-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013940100

Mathematics Subject Classification: 62H99 (general multivariate methods), 62J05 (linear regression), 62P12 (environmental applications)

Acknowledgments

We would like to thank Regina van den Boogaart for her infinite patience and support to the authors during many periods of hard work and Matevz Bren for his help with the package and the fruitful collaboration on the special compositions. We are especially indebted to Vera Pawlowsky-Glahn and Juan-Jose Egozcue for a long running general support, without which none of us would be writing a book on compositional data analysis.

Contents

List of Figures

Chapter 1
Introduction

Abstract Data describing amounts of components of specimens are compositional if the size of each specimen is constant or irrelevant. Ideally compositional data is given by relative portions summing up to 1 or 100 %. But more often compositional data appear disguised in several ways: different components might be reported in different physical units, different cases might sum up to different totals, and almost never all relevant components are reported. Nevertheless, the constraints of constant sum and relative meaning of the portions have important implications for their statistical analysis, contradicting the typical assumptions of usual uni- and multivariate statistical methods and thus rendering their direct application spurious. A comprehensive statistical methodology, based on a vector space structure of the mathematical simplex, has only been developed very recently, and several software packages are now available to treat compositional data within it. This book is at the same time a textbook on compositional data analysis from a modern perspective and a sort of manual on the **R**-package "compositions": both **R** and "compositions" are available for download as free software. This chapter discusses the need of an own statistical methodology for compositions, the historic background of compositional data analysis, and the software needs for compositional data analysis.

1.1 What Are Compositional Data?

1.1.1 Compositions Are Portions of a Total

Traditionally, a dataset has been called *compositional* if it provides portions of a total: percentages of workers in different sectors, portions of the chemical elements in a mineral, concentration of different cell types in a patient's blood, portions of species in an ecosystem or in a trap, concentration of nutrients in a beverage, portions of working time spent on different tasks, portions of types of failures, percentages of votes for political parties, etc.

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_1, © Springer-Verlag Berlin Heidelberg 2013

The individual parts of the composition are called *components*. Each component has an *amount*, representing its importance within the whole. Amounts can be measured as absolute values, in amount-type physical values like money, time, volume, mass, energy, molecules, individuals, and events. Or they can be also measured as portions with respect to a total, simply reporting such a quantity divided by a common total.

The sum over the amounts of all components is called the *total amount* or, short, the *total*. *Portions* are the individual amounts divided by this total amount. Depending on the unit chosen for the amounts, the actual portions of the parts in a total can be different. For instance, different numbers for mass percentage, volume percentage, and molar percentage will be obtained for the same physical system.

Most methods from multivariate statistics developed for real valued datasets are misleading or inapplicable for compositional datasets, for various reasons:

- *Independent components mixed together and closed exhibit negative correlations (Chayes, 1960).*
  This *negative bias* contradicts the usual interpretations of correlation and covariance, where independence is usually related to zero correlation.
- *Covariance between two components depends on which other components are reported in the dataset.*
  This disqualifies classical covariance-based tools as objective descriptions of the dependence between just two variables. This is particularly important whenever there is no single, unique, objective choice of the components to include in the composition, because each analyst could reasonably take a different set of components (including the two for which the covariance is computed) and each would get a different covariance estimate, even with different signs. The same comment applies to correlation: this is known as the *spurious correlation* problem.
- *Variance matrices are always singular due to the constant sum constraints.*
  Many multivariate statistical methods rely on a full-rank variance matrix, like multivariate linear models, Mahalanobis distances, factor analysis, minimum determinant variance estimators, multivariate Z-transforms, multivariate densities, linear and quadratic discriminant analysis, and Hotelling's T^2 distribution. None will be directly applicable.
- *Components cannot be normally distributed, due to the bounded range of values.*
  Many multivariate methods are at least motivated by multivariate normal distributions: covariance matrices, confidence ellipsoids, and principal component analysis are some examples. The normal model is a bad one for (untransformed) compositions, because it is a model unable to describe bounded data.

Compositions thus need an own set of statistical methods and should not be treated with statistical methods made for interval scale data. That early realization motivated Aitchison (1986) to introduce the field of compositional data analysis.

1.1.2 Compositions Are Multivariate by Nature

It is quite evident that our dataset can only be compositional if it has at least two components. Otherwise, we cannot speak of *a part in a total*. Even in the case that we only report one part, we are in fact implicitly relating it either to a predefined total or to a complementary part. For instance, the votes obtained by the democrats/labor/social party in a district are uninformative without knowing how many got the republicans/tories/conservatives or how many votes were counted altogether; the number of grains of feldspar counted on a sediment must be interpreted relative to the grains of quartz or to the total number of grains in the counting procedure; the investment of a state on Education is on itself meaningless without knowing either how large was the total budget or how much does it spend on defense.

That implies a substantial difference between compositional data and other multivariate datasets. Most multivariate analysis begin with an univariate analysis of the individual variables (the *marginals*), whereas each marginal variable of a compositional dataset has no meaning on itself, isolated from the rest.

1.1.3 The Total Sum of a Composition Is Irrelevant

Classically, compositions were defined as vectors of positive components and constant sum. However, in a more recent perspective (Aitchison, 1997; Barceló-Vidal, 2000; Barcelo-Vidal, 2003), a dataset of amounts of components of a common total can (and should) be modeled as *compositional* if the total amount does not matter for the problem under consideration. This may occur for several reasons:

- *When compositional data are actually measured, we often have little control over the total amount.*
  A probe of the patient's blood, a sample of geological formation, a national economy, or the workforce of a company: in any of these cases, the total size of each individual sample is either irrelevant (the country population, the size of the rock sample, or the enterprise) or predefined (the syringe volume).
- *Often the amounts are incompletely given and do not sum up to the "real" total.*
  Some species of a biological system cannot be caught in traps; we never quantify all possible chemical elements in a rock; we cannot analyze for all possible ingredients of a beverage.
- *Most typically, the totals are not comparable between the different statistical individuals.*
  It is well known that different countries have different population, that rock samples are parts of larger formation bodies, that smaller patients have less blood, that the Amazonas carries more water than the Rhone, or that beverages might be sold in different bottles. However, as the size of the bottle is irrelevant for studying the recipe, the size of the country is irrelevant when characterizing its

economy as rural or industrial. Whereas factors explaining the total might vary (history, sample preparation, trapping efficiency, patient's weight, or geography) and are very relevant for many problems, the composition itself is typically only indirectly related to the total size.

By taking compositions as vectors with irrelevant total sum, we get a more inclusive definition, since it does not depend on the formal criterion of a constant sum. It rather relies on a modeling choice based on the scientific question. Any dataset that, if completed, may meaningfully fit into the total-sum definition can be treated as compositional. Any set of amounts of components that constitute parts of a total can thus be called a composition.

In this new approach the different components might be measured in different units like μg, g, oz, or lbs for mass. Each component, i.e. each column of the dataset, has to be measured in a single physical value and unit throughout the dataset. One compositional dataset should only use proportional physical values; e.g., amounts mol and g might freely be mixed as long as the molar masses of the material given in mol are constant. It is not necessary to know the molar mass. Under these mild conditions the principles of compositional analysis discussed in Sect. 2.2 will ensure consistency of the results. However, for a meaningful interpretation of totals and individual portions, all measurements need to be transformed into common units.

Note that, if the total amount varies and we have missing data, i.e., some individual components could not be observed, it is even impossible to obtain portions or percentages of the non-missing variables, because the total cannot be computed. The data may nevertheless be still compositional, as long as the (non-computable) total is irrelevant for our scientific problem.

1.1.4 Practically All Compositions Are Subcompositions

If a composition A contains only parts of another composition B, but not necessarily all of them, A is called a *subcomposition* of B (Aitchison, 1986).

In fact, compositional data almost never contains all possible parts: many employment studies forget about housewives, unemployed, students, retired, or illegals; we do not measure water and "unimportant" trace elements in a hydrochemical analysis; we might not consider plasma or platelets when analyzing components in a blood probe; we neither catch mammals nor flies in traps; we never measure cellulose in beverages; we ignore breaks and socialization when analyzing work time patterns; we ignore the non-failure or "user-stopped-trying-failure" in failure analysis; and we ignore nonvoters along with minor parties in election statistics. Practically all compositions can thus be considered as a subcomposition of a larger composition describing a more complete reality.

Therefore, Aitchison (1986) demands that any analysis of compositional data should be executed in a way that the results on a subset of components do not depend on the presence or absence of other irrelevant components in the dataset.

This principle is called *subcompositional coherence*. Almost none of the classical multivariate methods satisfies this condition when directly applied to compositions.

1.1.5 The Very Brief History of Compositional Data Analysis

Awareness on the fact that the singular variance matrices of proportions make multivariate statistical analysis yield uninterpretable results arose in the field of Geology in the second half of the twentieth century. Starting with the famous paper of Chayes (1960) on spurious correlation, a series of papers bloomed, where the pathology of spurious correlation was diagnosed in several classical multivariate methods (e.g., Butler, 1978, 1979; Chayes and Trochimczyk, 1978; Pawlowsky-Glahn, 1984). Surprisingly, such problems remained fairly unnoticed by scientists from the many other fields of science where compositional data appear, even though a much earlier warning to statisticians can be traced back to Pearson (1897).

In a seminal work Aitchison (1986) developed an axiomatic approach to compositional data analysis with a variety of methods, operations, and tools, based on a set of fundamental principles, like subcompositional coherence. Its grounding idea was that compositions should be always treated in a log-ratio-transformed scale. Unfortunately, the mathematical complexity, the fundamental difference to well-known multivariate statistics, and especially some inertia of the analysts have rendered the subject fairly unknown to many scientists for a long time.

Three milestones can be identified in the way to the present better general understanding of compositional data analysis. First, Aitchison (1997) and afterwards Barceló-Vidal (2000) with much grounding presented compositions as equivalence classes, suggesting that any analysis results should not change if we scale our compositions by arbitrary positive numbers. Next, Pawlowsky-Glahn and Egozcue (2001) and Billheimer et al. (2001) independently realized that the fundamental compositional operations of Aitchison define a vector space on the simplex. Finally, Pawlowsky-Glahn (2003) formulated the principle of working in coordinates, which directly allows for a fast and transparent translation of most classical multivariate methods to deal with compositional data.

The principle of working in coordinates states that classical multivariate methods should be applied to (additive, centered or) isometric log-ratio-transformed versions of the data and that the results can be meaningfully back-transformed into compositions when interpreting the results. Which transformation is best used depends on the type of analysis. This understanding allows to handle compositional data in analogy to classical multivariate real data. Compositional methods are now widely available, easy to understand, and provide answers for a wide range of statistical problems.

In the meantime, some abstract consequences of the Aitchison (1986) approach to compositional data analysis were rediscussed in the light of some particular applications. That sometimes yield compositional data analysis conflict with other basic principles of the application, such as mass conservation (Shurtz, 2003), or

confront classical interpretations (Cortes, 2009). We should thus conclude that the Aitchison approach, although nowadays the most well-founded one, is not the only way to look at compositional data.

1.1.6 Software for Compositional Data Analysis

Three comprehensive software packages for compositional data are available:

- **CODA**; A microcomputer package for the statistical analysis of composition data by Aitchison (1986).
  A collection of programs running under Microsoft-DOS, also ported to Matlab.
- **CoDaPack3D** Thió-Henestrosa et al. (2003).
  An Excel and Visual Basic based software for compositional data analysis. An Excel-free version has been recently made available.
- **Compositions** (van den Boogaart et al., 2009).
  An **R**-package for compositional data analysis.

While the first package of Aitchison from 1986 can be fairly considered as outdated, *CoDaPack* is a good platform for a basic compositional analysis for Excel users who prefer a menu-based interface. This book is based on the "`compositions`" package and general tools for multivariate analysis available in **R** (R Development Core Team, 2004), which allow a more flexible analysis. The book provides with the necessary theoretical background and the practical skills to perform an elementary or advanced compositional data analysis within **R**.

1.2 Getting Started with R

1.2.1 Software Needed for the Examples in This Book

All examples in this book are computed with the statistical computing language **R** (R Development Core Team, 2004) using several extension packages. The examples are intended to allow the reader to apply similar methods to his own data. Obviously, **R** itself and "`compositions`" (in its December 2010 version) must be installed in the PC before you replay the examples. "`compositions`" depends on the "`rgl`" package for 3D drawing in 3D quaternary diagrams and the "`tensorA`" package (van den Boogaart, 2007) for tensorial computation in estimation under the presence of missing values. Moreover, the following packages are needed in some of the examples: "`colorspace`" (Ihaka et al., 2009) to get color ramps, "`robustbase`" (Rousseeuw et al., 2007) for robust linear model fitting: "`energy`" (Rizzo and Szekely, 2008) and "`cramer`" (Franz, 2006) for some non-parametric tests, "`combinat`" (Chasalow, 2005) to work with the multinomial distribution, and

either "MCMCpack" (Martin et al., 2008) or "gtools" (Warnes, 2008) to work with the Dirichlet density. Eventually it might be necessary to download example datasets and further hints from the home page of the book.[1] This page also contains the **R** code used in this book.

1.2.2 Installing R and the Extension Packages

To install **R** one typically downloads the "binary distribution" of the "R base" for the operative system used from the comprehensive **R** archive network CRAN[2]. There is a "Readme.html" file with installation instructions for each OS on the system-specific download pages.

Users of Windows and Mac-OS can download and install the contributed packages by a menu command, from within the **R** GUI, whenever the computer is connected to the Internet.

Users of Unix/Linux download the source package of the necessary contributed extension packages manually from CRAN and install them by executing the command

```
R CMD INSTALL completenameofpackagefile
```

in a root shell, or by

```
sudo R CMD INSTALL completenameofpackagefile
```

Alternatively, packages can also be installed with the command `install.packages` from an **R** session with administrator rights. Also, some Linux distributions include many packages in their own repositories that should be preferentially used.

Unix/Linux users can start **R** with the simple command "`R`" on a console. Users of Windows and Mac find the program `RGui`, where typical for their configurations, in the start menu or the dock bar.

More detailed installation instructions and other manuals for **R** are available on **R** home page http://cran.r-project.org/.

1.2.3 Basic R Usage

Although the book does not require any prior knowledge on **R**, it is best understood with some knowledge on it, which, e.g., can be learned from Dalgaard (2008) or from

```
http://www.cran.r-project.org/doc/manuals/R-intro.pdf.
```

[1] http://www.stat.boogaart.de/compositionsRBook.

[2] http://www.cran.r-project.org.

When **R** is started it displays—eventually in a new window or in the console—an **R**-console displaying a startup message and a prompt typically looking like:

```
>
```

All **R** commands in this book should be typed after that prompt. If the command is given on a separate line, it is typically displayed with the prompt. For example, we can perform some simple calculation with **R**:

```
> 1+(1+2)*5

[1] 16

> sin(0.25*pi)^2

[1] 0.5
```

The lines with a prompt `>` show the commands the user needs to type. The results show that **R** obeys the basic calculation precedence rules, knows basic functions like sin and constants like π and has a powering operator, and calculates trigonometric functions in radians, because $\sin((1/4)\pi) = \sqrt{1/2}$. Results of calculation can be assigned to new or existing variables by the `=` or the `<-` operator and reused in later calculations:

```
> a = 1+1
> a+1

[1] 3
```

Variables in **R** can contain numbers, strings, vectors, matrices, datasets, and many other types of objects. By default, a value assigned to a variable is not printed. However, we can force printing of the value by enclosing the assignment in `( )`:

```
> (a = 1+1)

[1] 2
```

We will use this forced printing extensively to follow the steps of calculations closely. **R** can assign a vector of values (concatenated by the `c` command) to a variable:

```
> (a = c(1,2,pi,exp(0)))

[1] 1.000 2.000 3.142 1.000
```

and often uses vectorized computation, in the sense that applying a function to the variable will compute the result for each entry:

```
> log(a)

[1] 0.0000 0.6931 1.1447 0.0000
```

```
> a+1

[1] 2.000 3.000 4.142 2.000

> a+a

[1] 2.000 4.000 6.283 2.000
```

One or more elements can be extracted from a vector by the `[]` index expression:

```
> a[3]

[1] 3.142

> a[c(2,4)]

[1] 2 1
```

The entries of the vectors can further be used as entries of a matrix:

```
> (A <- matrix(a,nrow=2))

     [,1]  [,2]
[1,]    1 3.142
[2,]    2 1.000
```

and then indexed by rows and columns, where a "," is used to separate row and column index:

```
> A[1,2]

[1] 3.142

> A[1,]

[1] 1.000 3.142

> A[,1]

[1] 1 2

> A[,c(2,1)]

      [,1] [,2]
[1,] 3.142    1
[2,] 1.000    2
```

In **R** rows and columns can have names, vectors can contain character strings as entries, and datasets (`data.frame`) might even contain different data types for each column. All these concepts will naturally appear at some places in this book: hopefully, the context will be clear enough to allow the reader to follow the explanations.

The reader may notice that the code in this book is executed with a limitation in the output length, for the sake of clarity and to save space. This was obtained with the settings

```
> options(width=65,digits=4)
```

ensuring that line breaks in the output correspond to the width of the book and reduce the number of digits. We strongly encourage the readers to set the same settings in their **R** sessions.

Having a contributed package installed does not make **R** directly aware of it: we have to load it in the memory. This is done with the commands `library` or `require`, giving the name of the desired package as argument. Windows and Mac users also have a menu option to load packages.

```
> require(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 :

     %*%,
     scale,
     scale.default
```

The output tells us that the package loads a set of additional packages and modifies some standard commands like correlations `cor` or inner products `%*%`. This is done to augment the functionality of these commands for a specialized treatment of compositional data, by turning them into generic commands ("`compositions`" currently uses S3-generics).

1.2.4 Getting Help for Composition Specific Commands

Generic commands are able to execute different code when applied to different types of objects. To find help for such specialized commands, one first needs to find out the actual name of the command used for a given type of data, by finding the different methods available, with `methods`. For instance, if we want to see what happens when we compute the variance of a compositional dataset, we should first look for the methods available for `var`,

```
> methods(var)
```

```
 [1] var.acomp   var.aplus   var.default var.linear  var.lm
 [6] var.mlm     var.rcomp   var.rmult   var.rplus   var.tensor
[11] var.test
```

and then request help for the class of data used; e.g., compositional data should be stored as a `acomp` (for Aitchison COMPosition) object, and thus we would get the explanation of the compositional variance with a `help` command like

```
> ? var.acomp
```

1.2.5 Troubleshooting

When a command or code chunk from this book does not work for you, check the following things:

- Have earlier commands on the same dataset been executed for computing intermediate results? Did you try something else between, eventually overwriting them? Then, find out which variables did you forget or overwrite, and copy again the code lines where they were defined.
- Is the "`compositions`" package installed and *loaded*? Are other accessory contributed packages also loaded? Load the missing package again with `library(missing.package.name)`.
- Are the commands typed correctly, e.g., is that a 1 ("one") or an l ("el")?
- Are referenced files present and the corresponding file paths adapted, using the path separator "/" rather than "\"? **R** uses the path notation conventions of Unix, not those of Windows. To use "\" in strings, type "\\" instead.
- Are variable names adapted to your dataset?
- Maybe there are changes in the newer version of the software you are using. Check the home page[3] of the book for updated versions of the commands.

References

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. London: Chapman & Hall (Reprinted in 2003 with additional material by The Blackburn Press), 416 pp.

Aitchison, J. (1997). The one-hour course in compositional data analysis or compositional data analysis is simple. In V. Pawlowsky-Glahn (Ed.), *Proceedings of IAMG'97—The third annual conference of the International Association for Mathematical Geology*, Volume I, II and addendum (pp. 3–35). Barcelona: International Center for Numerical Methods in Engineering (CIMNE), 1100 pp.

Barceló-Vidal, C. (2000). *Fundamentación matemática del análisis de datos composicionales*. Technical Report IMA 00-02-RR. Spain: Departament d'Informática i Matemática Aplicada, Universitat de Girona, 77 pp.

Barcelo-Vidal, C. (2003). When a data set can be considered compositional? See Thió-Henestrosa and Martín-Fernández (2003).

[3] http://www.stat.boogaart.de/compositionsRBook.

Billheimer, D., Guttorp, P., & Fagan, W. (2001). Statistical interpretation of species composition. *Journal of the American Statistical Association, 96*(456), 1205–1214.

Butler, J. C. (1978). Visual bias in R-mode dendrograms due to the effect of closure. *Mathematical Geology, 10*(2), 243–252.

Butler, J. C. (1979). The effects of closure on the moments of a distribution. *Mathematical Geology, 11*(1), 75–84.

Chasalow, S. (2005). *combinat: Combinatorics utilities*. R package version 0.0-6.

Chayes, F. (1960). On correlation between variables of constant sum. *Journal of Geophysical Research, 65*(12), 4185–4193.

Chayes, F., & Trochimczyk, J. (1978). An effect of closure on the structure of principal components. *Mathematical Geology, 10*(4), 323–333.

Cortes, J. A. (2009). On the harker variation diagrams; a comment on "the statistical analysis of compositional data. Where are we and where should we be heading?" by Aitchison and Egozcue (2005). *Mathematical Geosciences, 41*(7), 817–828.

Dalgaard, P. (2008). *Introductory tatistics with R* (2nd ed.). Statistics and computing. Berlin: Springer.

Franz, C. (2006). *cramer: Multivariate nonparametric Cramer-Test for the two-sample-problem*. R package version 0.8-1.

Ihaka, R., Murrell, P., Hornik, K., & Zeileis, A. (2009). *colorspace: Color Space Manipulation*. R package version 1.0-1.

Martin, A. D., Quinn, K. M., & Park, J. H. (2008). *MCMCpack: Markov chain Monte Carlo (MCMC) Package*. R package version 0.9-4.

Pawlowsky-Glahn, V. (1984). On spurious spatial covariance between variables of constant sum. *Science de la Terre, Série Informatique, 21*, 107–113.

Pawlowsky-Glahn, V. (2003). Statistical modelling on coordinates. See Thió-Henestrosa and Martín-Fernández (2003).

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

Pearson, K. (1897). Mathematical contributions to the theory of evolution. On a form of spurious correlation which may arise when indices are used in the measurement of organs. *Proceedings of the Royal Society of London, LX*, 489–502.

R Development Core Team (2004). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-00-3.

Rizzo, M. L., & Szekely, G. J. (2008). *energy: E-statistics (energy statistics)*. R package version 1.1-0.

Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Maechler, M., et al. (2007). *robustbase: Basic Robust Statistics*. R package version 0.2-8.

Shurtz, R. F. (2003). Compositional geometry and mass conservation. *Mathematical Geology, 35*(8), 927–937.

Thió-Henestrosa, S., Barceló-Vidal, C., Martín-Fernández, J., & Pawlowsky-Glahn, V. (2003). CoDaPack. An Excel and Visual Basic based software for compositional data analysis. Current version and discussion for upcoming versions. See Thió-Henestrosa and Martín-Fernández (2003).

Thió-Henestrosa, S., & Martín-Fernández, J. A. (Eds.). (2003). *Compositional data analysis workshop—CoDaWork'03, Proceedings*. Universitat de Girona, ISBN 84-8458-111-X, http://ima.udg.es/Activitats/CoDaWork03/.

van den Boogaart, K. G. (2007). *tensorA: Advanced tensors arithmetic with named indices*. R package version 0.31.

van den Boogaart, K. G., Tolosana, R., & Bren, M. (2009). *compositions: Compositional data analysis*. R package version 1.02-1.

Warnes, G. R. (2008). *gtools: Various R programming tools*. Includes R source code and/or documentation contributed by Ben Bolker and Thomas Lumley.

Chapter 2
Fundamental Concepts of Compositional Data Analysis

Abstract Compositional data is considered a statistical scale in its own right, with its own natural geometry and its own vector space structure. Compositional data analysis and this book cannot be understood without a basic knowledge of these issues and how they are represented in **R**. Therefore, this chapter introduces the basic geometric concepts of compositional data and of the **R**-package "`compositions`": the relative nature of compositional data; how to load, represent, and display compositional data in **R**; the various compositional scales and geometries and how to select the right geometry for the problem at hand; and how to specify the geometry in "`compositions`" using the basic classes "`acomp`", "`rcomp`", "`aplus`", "`rplus`", "`ccomp`", and "`rplus`". A concise guide to the most important geometry for compositional data, the Aitchison geometry, is also included. The whole book relies on these basic principles, and the reader should make him or herself familiar with them in Sects. 2.1–2.5 before going on.

2.1 A Practical View to Compositional Concepts

2.1.1 Definition of Compositional Data

A *composition* is often defined as a vector of D positive components $\mathbf{x} = [x_1, \ldots, x_D]$ summing up to a given constant κ, set typically equal to 1 (portions), 100 (percentages), or 10^6 (ppm). We already mentioned that this definition is misleading, because many compositional datasets do not *apparently* satisfy it. Let us see how.

Consider, for instance, this simple example of compositional data collected from typical nutrition tables of foodstuff in any kitchen cupboard:

```
      Fat  Sodium  Carbonates Protein  Total
soy     7      14          10      13    100
peas    3      12          61      22    100
```

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_2, © Springer-Verlag Berlin Heidelberg 2013

```
  wheat     1         15          60       23     100
  corn      1        180          15        2      82
  beans     0.5      680          30       12      NA
```

This data can be loaded from a text file[1] into **R** with the command `read.table`:

```
> library(compositions)
> kitchen <- read.table("SimpleData.txt",sep="")
```

All the amounts are given in grams g, except for sodium which is given in milligrams mg. A naive interpretation of these raw numbers would tell us that soy and corn are poor in protein and carbonates, whereas beans have seemingly 4 times more sodium and twice more carbonates than corn. However, this naive (i.e., non-compositional) direct interpretation contradicts everything we know about soy or corn. For a good compositional analysis, it is important to understand the fundamental operations and laws governing compositional analysis to avoid these pitfalls. This chapter introduces this methodological background step by step.

2.1.2 Subcompositions and the Closure Operation

A first inspection shows that the entries do not sum up to the total 100 %, since these products do not exclusively contain the nutrients reported. We thus only get some parts of a bigger composition, which would include water, vitamins, and further minerals. For instance, these non-reported components represent $100-(7+14/1000+10+13)\sim 70$ g out of 100 g of soy.

As people with a cholesterol problem and an office job, we may be interested in getting the most protein for the least amount of fat while maintaining a reasonable medium amount of carbohydrates. Our water and sodium intake is much more influenced by drinking and salting than by a diet choice between beans and peas. Thus, we are going to focus our attention to the composition of fat–protein–carbohydrates. It is typical to *close* the amounts in the parts of interest to sum up to 100 %. In "`compositions`," this can be done with the command `clo` (for close composition):

```
> (coi <- clo(kitchen,parts=
              c("Fat","Carbonates","Protein"),total=100))

         Fat Carbonates Protein
soy   23.333      33.33   43.33
peas   3.488      70.93   25.58
wheat  1.190      71.43   27.38
```

[1]The "`SimpleData.txt`" file can be downloaded from http://www.stat.boogaart.de/compositionsRBook.

```
corn   5.556      83.33   11.11
beans  1.176      70.59   28.24
```

These are now portions given in mass %. The command `clo` has an optional parameter `parts` taking the names or the numbers of columns to be selected and an optional argument `total` specifying the new reference total. If `parts` is not specified, implicitly all columns are used. If `total` is not specified, a default of 1 is assumed.

Note that a naive interpretation of these numbers would now tell us that soy is the richest in protein (whereas before, it was rather in the middle of the ranking for protein amount), and that corn is now richer in carbonates than beans. This illustrates the main risk of interpreting raw portions: our conclusions can be opposite, depending on which composition we are observing.

A composition only representing some of the possible components is called a *subcomposition*. Most of real compositional data is actually representing a subcomposition, as we never analyze each and every possible component of our samples. Aitchison (1986) therefore introduced the principle of *subcompositional coherence*: any compositional data analysis should be done in a way that we obtain the same results in a subcomposition, regardless of whether we analyzed only that subcomposition or a larger composition containing other parts. The interpretation we have been doing of the kitchen example up to now was not coherent, as our consideration of which food is richer in carbonates or protein changes between the original composition and the subcomposition of interest. This concern affects the whole direct application of all traditional multivariate statistics: none obey the principle of subcompositional coherence.

Reclosure is often a controversial *manipulation*, as by closing a subcomposition, it seems that we lose some information on the total mass present in the parts considered, and *apparently* we replace some measured values by some computed ones. However, we must consider that some of the cases in the kitchen dataset are dry and other cooked conserves, some even pre-salted, such that I do not need to add salt anymore. Some of the sodium (and of the other components, as well as water content) are thus artifacts of the form of delivery: they are related neither to the crop properties nor to the final cooked product we might eat. The only relevant quantities unaltered throughout these manipulations are the ratios between nutrients, and they are preserved by the closure operation.

2.1.3 Completion of a Composition

An alternative way to recast a vector of amounts to a composition is by adding the complementary variable. We first select the compositional variables using usual vectorized indexing[2] with `[]` :

[2]Call `help("[")` in **R** for a detailed explanation.

```
> amounts <- kitchen[,c("Fat","Carbonates","Protein")]
> amounts
```

```
       Fat Carbonates Protein
soy    7.0         10      13
peas   3.0         61      22
wheat  1.0         60      23
corn   1.0         15       2
beans  0.5         30      12
```

and then compute the total grams of each column by

```
> (sumPresent <- totals(rplus(amounts)))
```

```
  soy  peas wheat  corn beans
 30.0  86.0  84.0  18.0  42.5
```

This last line marks the dataset as positive real numbers by applying the `rplus` class[3] and then computes the sum of the individual amounts, by rows with the command `totals`.

We can now create a new dataset with an additional column `Other` giving the remaining mass:

```
> Other <- kitchen["total"] - sumPresent
> Amounts <- cbind(amounts,Other=Other)
> clo(Amounts)
```

```
           Fat Carbonates Protein  Other
soy    0.07000     0.1000 0.13000 0.7000
peas   0.03000     0.6100 0.22000 0.1400
wheat  0.01000     0.6000 0.23000 0.1600
corn   0.01220     0.1829 0.02439 0.7805
beans  0.01176     0.7059 0.28235     NA
```

The `cbind` command (for *c*olumn *bind*) binds the columns of its arguments to a joint dataset containing all the columns.

Of course, this procedure is only applicable when we know how much sample was analyzed in order to subtract from it the sum of the present components. This completion procedure has an advantage over the reclosure of the observed subcomposition: the reclosure loses all information about how much we had in total of the observed components, whereas in the completion procedure this total present is inherited by the "Other" part.

A look at the last line reveals a practical problem of both the closure and the completion procedures in the presence of missing values. To close a composition with missing elements, we need to guess a value for the missing elements. The computer cannot add up unknown values and thus will close the rest of the

[3]The different compositional classes will be discussed in Sect. 2.4.

composition to 1, such as if this last components would be absent. However, it is very unlikely that the canned beans do not contain a considerable amount of water.

2.1.4 Compositions as Equivalence Classes

Either by reclosing a subcomposition or by completing the composition, the dataset is now compositional in nature, since its columns sum up to a constant. In any way, we homogenized the totals to the same quantity, that is to say: we removed the information on the total amount of each sample. This is licit, because that total is in our kitchen example not relevant to evaluate the nutritive character of the different crops: it is an arbitrary choice of the producer or due to the local unit system. In fact, we could rescale each row of the dataset by multiplying it with a positive value, and the information they convey would not change. We can thus say that two vectors are *compositionally equivalent* (Aitchison, 1997; Barceló-Vidal, 2000) if one is simply some multiple of the other:

$$\mathbf{a} =_A \mathbf{b} \Leftrightarrow \text{ exists } s > 0 \text{ for all } i : a_i = sb_i$$

In other words, all vectors with positive entries that are scaled versions of one another represent the same composition. For instance, the following three vectors are the same composition

```
> Amounts[4,]*100/82

      Fat Carbonates Protein Other
corn 1.22      18.29   2.439 78.05

> Amounts[4,]

     Fat Carbonates Protein Other
corn   1         15       2    64

> acomp(Amounts[4,])

     Fat    Carbonates Protein Other
corn 0.0122 0.1829     0.02439 0.7805
attr(,"class")
[1] acomp
```

first giving the composition of corn as portions of 100 g or mass %, then as portions of 82 g, and finally, as relative portions of 1. Note that in the last case, we used the command `acomp`, which tells the system to consider its argument as a compositional dataset, implicitly forcing a closure to 1. The different compositional classes (like `acomp`, or `rplus` before) are the grounding blocks of "`compositions`" and will be discussed in detail in Sect. 2.4.

2.1.5 Perturbation as a Change of Units

Indeed mass percentage is a quite irrelevant type of quantification for us: we are more interested in the energy intake. But the different nutrients have a different energy content (in kJ/g):

```
> (epm <- acomp(c(Fat=37,Protein=17,Carbonates=17)))

       Fat    Protein Carbonates
    0.5211     0.2394     0.2394
attr(,"class")
[1] acomp
```

Since our dataset is not anymore in grams but scaled to unit sum, it is likewise irrelevant in which units the energy contribution is given: the energy per mass can actually be seen again as a composition.

We can now transform our mass percentage subcomposition *coi* of nutrients to a energy composition *ec* by scaling each entry with the corresponding energy per mass composition and reclosing. For example, for the first line of the dataset:

$$\begin{aligned}(ec_{1i})_{i=1,\dots,n} &= \mathscr{C}(coi_{1i}\, epm_i)_{i=1,\dots,n} \\ &= \frac{(23.33 \cdot 0.52, 33.33 \cdot 0.24, 43.33 \cdot 0.24)^t}{23.33 \cdot 0.52 + 33.33 \cdot 0.24 + 43.33 \cdot 0.24} \\ &= (0.4, 0.26, 0.34)^t\end{aligned}$$

This operation is called *perturbation*. It is denoted by a $\oplus$ sign:

$$ec = coi \oplus epm$$

In R, it is done simply by *adding* two acomp objects:

```
> (ec <- acomp(coi)+epm)

      Fat     Carbonates Protein
soy   0.39846 0.2615     0.3400
peas  0.07293 0.6813     0.2457
wheat 0.02555 0.7044     0.2700
corn  0.11350 0.7822     0.1043
beans 0.02526 0.6962     0.2785
attr(,"class")
[1] acomp
```

Now, `ec` contains the composition of my ingredients measured in terms of portion of energy rather than in mass. Perturbation is considered as the fundamental operation in compositional data analysis, equally important as the vector addition is in the usual real vector space $\mathbb{R}^D$. In an abstract mathematical view, perturbation is

indeed the *addition* in a vector space structure. Section 2.5 formally introduces this operation and the properties of this vector space structure.

2.1.6 Amalgamation

A typical operation when dealing with compositional data is to amalgamate some of the variables into a single component. For instance, in the kitchen data reported before, fat was given as a single component. Though the original information gave the proportions of saturated and unsaturated fats,

```
> fatDataset
          total.Fat  sat.Fat     unsat.Fat
soy       7          1.40        5.60
peas      3          0.60        2.40
wheat     1          0.25        0.75
corn      1          0.32        0.68
beans     0.5        0.10        0.10
```

we amalgamated them into the total fats. Amalgamation, though very commonly done, is a quite dangerous manipulation: a fair amount of information is lost in a way that we may find ourselves unable to further work with the amalgamated dataset. For instance, if saturated and unsaturated fats have different energy content, we cannot compute the total fat mass proportion from the total fat mass energy content, or vice-versa. Amalgamation thus should only be applied in the "definition of the problem" stage, when choosing which variables will be considered and in which units. It should be meaningful in the units we are dealing with, because afterwards, we will not be able to change them. And it should have a deep connection with our questions: to amalgamate apples and pears may be meaningful when investigating the nutrition habits of one population (where an apple may be equivalent to a pear) and catastrophic when studying the importance of some fruits in several cultures (where we just want to see different patterns of preference of fruits). Once the parts are defined, the units chosen, and the questions posed, we should not amalgamate a variable anymore.

The easiest way to amalgamate data in **R** is to explicitly compute the amalgamated components as totals of an unclosed amount dataset (of an amount class, `"rplus"` or `" aplus"`) containing only these components:

```
> fatDataset$total.Fat = totals(aplus(fatDataset,
                          c("sat.Fat","unsat.Fat")))#$
```

The `aplus` command works in the same way as `acomp` but explicitly states that the total is still relevant and no reclosure should be performed. It is used to assign the scale of a dataset of amounts in relative geometry. Amalgamation is further explained in Sect. 3.3.1.

2.1.7 Missing Values and Outliers

Note that the total amount for beans is missing (the nutrition table was reported for a 250 mL can of this product). We could have more missing values in the dataset by considering potassium, which is only given for some of the products, probably those for which it is relevant. However, it would be naive to assume that there is no potassium in any of these products.

Much more than in classical multivariate analysis, missing values in compositional data need a careful treatment, especially because a missing value in a single component makes it impossible to close or complete the data, such that finally, the actual portion of no single component is known. It will be thus very important to understand which procedures are still valid in the presence of missing values and how others might be (partially) mended. The same applies to many other kinds of irregularities in compositional data, like measurements with errors, atypical values, or values below the detection limit: in all these cases, the total sum will either be unknown or affected by an error, which will propagate to all the variables by the closure. The solution comes by realizing that whichever this total value might be, the (log)ratios between any regular components are unaffected.

The "`compositions`" package will thus close the non-missing parts as if the missing parts are 0, knowing that this value will not affect any proper analysis.

2.2 Principles of Compositional Analysis

Any objective statistical methodology should give equivalent results, when applied to two datasets which only differ by irrelevant details. The following four sections present four manipulations that should be irrelevant for compositions and that thus generate four invariance principles, with far-ranging consequences for the mathematics underpinning compositional data analysis (Aitchison, 1986).

2.2.1 Scaling Invariance

For the theoretical approach taken in this book, we will consider a vector as a composition whenever its components *represent the relative weight or importance of a set of parts forming a whole*. In this case, the *size* or total weight of that whole is irrelevant. One can then remove that apparent influence by forcing the data vectors to share the same total sum κ with the *closure operation*,

$$\mathscr{C}(\mathbf{x}) = \frac{\kappa \cdot \mathbf{x}}{\mathbf{1}^t \cdot \mathbf{x}}. \tag{2.1}$$

The components x_i of a closed composition $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ are values in the interval $(0, 1)$ and will be referred to as *portions* throughout this book. The term *proportions* will be reserved for ratios and relative sizes of components.

By applying the closure, we can compare data from samples of different sizes, e.g., the beans, corn, and wheat in our simple kitchen example (even though their compositions were respectively reported for 100 g, 82 g, and 250 mL of product); a sand/silt/clay composition of two sediment samples of 100 g and 500 g; and the proportion of votes to three parties in two electoral districts of 10^6 and 10^7 inhabitants. For instance, the vectors $\mathbf{a} = [12, 3, 4]$, $\mathbf{b} = [2400, 600, 800]$, $\mathbf{c} = [12/17, 3/17, 4/17]$, and $\mathbf{d} = [12/13, 3/13, 4/13]$ represent all the same composition, as the relative importance (the *ratios*) between their components is the same. A sensible compositional analysis should therefore provide the same answer independently of the value of κ or even independently of whether the closure was applied or else the data vectors sum up to different values. We say the analysis will be then *scaling invariant*. Aitchison (1986) already showed that *all* scaling-invariant functions of a composition can be expressed as functions of log ratios $\ln(x_i/x_j)$.

2.2.2 Perturbation Invariance

Compositional data can be presented in many different units, and even when given in portions, it is still relevant in which physical quantities the components were originally measured: g, tons, mass %, cm^3, vol. %, mols, molalities, ppm, ppb, partial pressure, energy, electric loading, money, time, persons, events, cells, mineral grains, etc. Even if it is clearly defined what substance we are measuring, how it is quantified is still a choice of the experimenter. It is extremely rare to find a composition where only one single type of quantification is meaningful: in most of the cases, several units could be equally chosen. And since different components are typically also qualitatively different, they might have different interesting properties. For instance, to change a mass proportion to a volume percentage, we need the densities of all components, which will be typically different. The same applies to changing prices between different valuations (from market price to book value, or to buying price, etc.), passing nutrition masses to energy intake, or cell type to energy consumption or biomass production. Thus, different analysts would end up having different values representing exactly the same compositions just due to a different choice of units.

It is evident that the statistical analyses we can apply should thus give the same qualitative results regardless of the units chosen, as long as they contain the same information, i.e., as long as we can transform the units into each other, e.g., by rescaling each component by its density. This operation is done by a perturbation with a composition containing as entries the conversion factors (e.g., the densities) for each component. We would thus require a meaningful compositional analysis to give the same results (obviously, up to a change of units) when applied to a dataset perturbed by this vector. But since we never know which type of quantifications might else be considered, we should request invariance of the analysis with respect to perturbation with all possible weighting factors.

This principle has the same level as the principle of translation invariance for real data. There, we would expect that, e.g., when we represent electric (or gravity) potentials with respect to a different reference level, we would still get the same results (up to that choice of reference level).

This principle is particularly critical when dealing with datasets of mixed units. Often different components are measured in different units. For instance, trace elements are often measured in ppm, while others are given in mg, or even kg, and fluids might be quantified in volume proportions or fugacities. Unfortunately, any method not honoring perturbation invariance would give completely arbitrary results when the units do not match. On the other hand, when we demand perturbation invariance, all such data could be meaningfully analyzed in a common framework, as long as there exists a perturbation bringing them into the same system of units, even if we do not know it.

2.2.3 *Subcompositional Coherence*

Sometimes, only a subset of the initial parts is useful for a particular application, and one works with the corresponding *subcomposition*, by reclosing the vector of the chosen components. Subcompositions play the same role with respect to compositions as marginals do in conventional real multivariate analysis: they represent subspaces of lower dimension where data can be projected for inspection. This has several implications jointly referred to as *subcompositional coherence* Aitchison (1986): three examples of these implications follow.

First, if we measure the distance between two D-part compositions, this must be higher when measured with all D components than when measured in any subcomposition.

Second, the total dispersion of a D-part compositional dataset must be higher than the dispersion in any subcomposition.

Finally, if we fitted a meaningful model to a D-part compositional dataset, the result should not change if we include a new non-informative (e.g., random) component and work with the resulting $(D + 1)$-part composition.

Remark 2.1 (Incoherence of classical covariance). The classical definitions of covariance and correlation coefficient are not subcompositionally coherent. This is connected to two problems: the *spurious correlation* and the *negative bias*, first identified by Pearson (1897) and later rediscovered by Chayes (1960). The spurious correlation problems states that the correlation between ratios with common denominators (e.g., two components of a closed composition) is arbitrary to an uncertain extent. The negative bias problem appears because each row or column of the covariance matrix of a closed composition sums up to zero: given that the variances are always positive, this implies that

some covariances are forced towards negative values, not due to any incompatibility process but because of the closure itself. For instance, if we compute the correlation coefficient between MnO and MgO content in a compositional dataset of glacial sediment geochemistry,[4] we may obtain 0.95 if we use the whole 10-part composition or −0.726 if we use only the subcomposition of elements not related to feldspar (P–Mn–Mg–Fe–Ti). Given the fundamental role of covariance in statistics, it is not a surprise that there exists a full series of papers on detecting, cataloging, and trying to circumvent the spurious correlation. An account can be found in Aitchison and Egozcue (2005).

2.2.4 Permutation Invariance

Last but not least, results of any analysis should not depend on the sequence in which the components are given in the dataset. This might seem self-evident, but it is surprising how many methods happen to violate it. For instance, one of the apparent bypasses of the constant sum of covariance matrices (Remark 2.1) was to remove the last component of the dataset: in that way, the dataset was not summing up to a constant anymore, and the covariance matrix was apparently free. That "solution" was obviously not permutation invariant (and even more, it was not a solution, as correlations had exactly the same values, being thus equally forced towards negative spurious correlations).

For the log-ratio approach, it is also a very important principle, when, e.g., we ask which methods can be meaningfully applied to coordinates of compositional data. For instance, a naive Euclidean distance of alr transformed data[5] is not permutation invariant and should thus not be used, e.g., for cluster analysis. We would otherwise risk to have different clusterings depending on which was the last variable in the dataset. This will be further discussed in Remark 2.2.

2.3 Elementary Compositional Graphics

Compositional datasets are typically represented in four different ways: as Harker diagrams (scatterplots of two components), as ternary diagrams (closed scatterplots of three components), as scatterplots of log ratios of several parts, or as sequences of bar plots or pie plots. By far, the most common ways are the first two.

[4] This dataset can be found on http://www.stat.boogaart.de/compositionsRBook.

[5] The alr (additive log ratio) transformation was the fundamental transformation in Aitchison (1986) approach, and it is discussed in detail in Sect. 2.5.7.

These compositional graphics will be illustrated with a dataset of geochemistry of glacial sediments (Tolosana-Delgado and von Eynatten, 2010). The dataset is available on the book home page[6]:

```
> GeoChemSed=read.csv("geochemsed.csv",
                      header=TRUE, skip=1)[,-c(3,14,31,32)]
> names(GeoChemSed)
```

```
 [1] "Sample" "GS"     "SiO2"   "TiO2"   "Al2O3"  "MnO"
 [7] "MgO"    "CaO"    "Na2O"   "K2O"    "P2O5"   "Fe2O3t"
[13] "Ba"     "Co"     "Cr"     "Cu"     "Ga"     "Nb"
[19] "Ni"     "Pb"     "Rb"     "Sc"     "Sr"     "V"
[25] "Y"      "Zn"     "Zr"     "Nd"
```

2.3.1 Sense and Nonsense of Scatterplots of Components

Harker diagram is the name given in geochemistry to a conventional scatterplot of two components, *without any transformation applied to them*. For this reason, they may highlight any additive relationship between the variables plotted: e.g., in the case of plotting two chemical components of a dataset evolving in several stages, Harker diagrams visually represent mass balance computations between the several stages (Cortes, 2009). Unfortunately, these diagrams are neither scaling nor perturbation invariant and not subcompositionally coherent (Aitchison and Egozcue, 2005): there is no guarantee that the plot of a closed subcomposition exhibits similar or even compatible patterns with the plot of the original dataset, even if the parts not included in the subcomposition are irrelevant for the process being studied. This is actually the spurious correlation problem; thus, a regression line drawn in such a plot cannot be trusted, in general terms. Figure 2.1 shows two Harker diagrams of the same components in the same dataset, exhibiting the subcompositional incoherence of this representation. One can obtain a Harker diagram by using the standard function `plot(x,y)`.

```
> par(mfrow=c(1,2), mar=c(4,4,0.5,0.5))
> plot( clo(GeoChemSed[,3:12])[,c(4,5)])
> plot( clo(GeoChemSed[,c(4,6,7,11,12)])[,c(2,3)])
```

2.3.2 Ternary Diagrams

Ternary diagrams are similar to scatterplots but display a closed three-part subcompositions. If we would plot the three components of a closed composition in a three-dimensional scatterplot, all points would lie in a planar triangle spanned

[6] http://www.stat.boogaart.de/compositionsRBook.

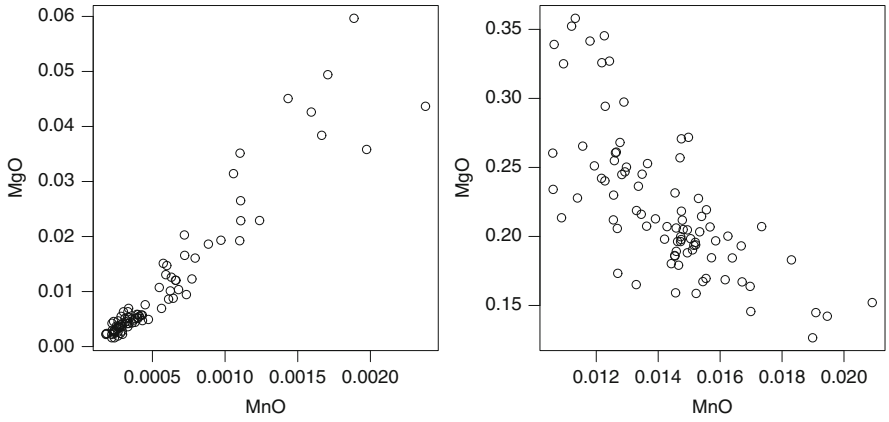


Fig. 2.1 Examples of Harker diagrams on the sediment geochemistry dataset. (*Left*) Using the full dataset of major oxides. (*Right*) Using the subcomposition P–Mn–Mg–Fe–Ti

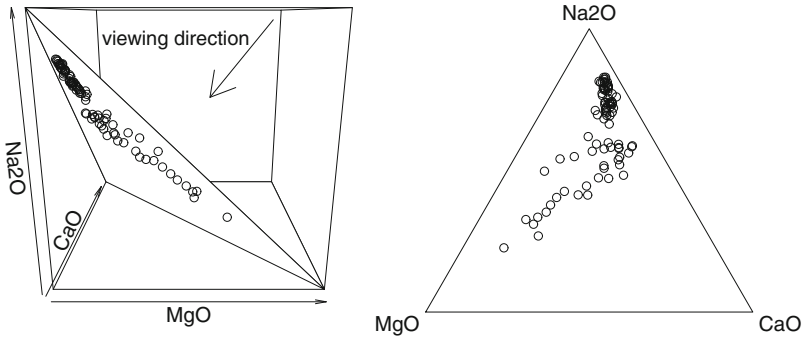


Fig. 2.2 Example of a ternary diagram embedded in its original three-dimensional space

by the three points $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ as corners. The ternary diagram displays this triangle in the drawing plane with the corners annotated by the axis on which this corner lies. This is possible since closed datasets of three parts have only 2 degrees of freedom. Figure 2.2 illustrates this with the MgO,CaO, and Na_2O subcomposition of the example dataset.

For the interpretation of ternary diagrams, we can make use of the property that the orthogonal segments joining a point with the three sides of an equilateral triangle (the *heights* of that point) have always the same total length: the length of each segment is taken as the proportion of a given part. Ternary diagrams have the merit of actually representing the data as what they are: compositional and relative. Geoscientists are particularly used to this kind of representation. Figures 2.3 and 2.4 show several ways in which the relative portions and proportions of the components of a three-part composition can be read from a ternary diagram.

It is worth mentioning that when the three parts represented have too different magnitudes, data tend to collapse on a border or a vertex, obscuring the structure:

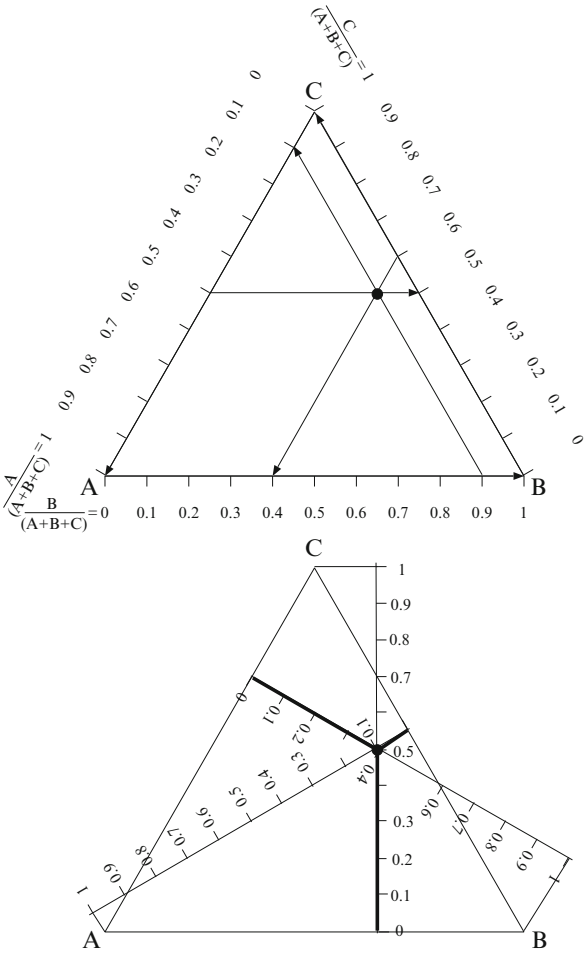


Fig. 2.3 In a ternary diagram, the portion of each part is represented by the portion of the height of the point over the side of the triangle opposite to the part on the total height of the triangle. However, it is difficult to judge height visually. If we assume the three sides of the triangle to form a sequence of axes, we can make a projection of the point onto the axis pointing towards the interesting part parallel to the previous axis. Consequently, a composition on an edge of the triangle only contains the two parts connected by the side, and a composition in a vertex of the triangle will only contain the part the vertex is linked to. Thus, all compositions along a line parallel to an axis have the same portion of the part opposite to the axis

a typical practice here is to multiply each part by a constant and reclose the result. One can obtain a ternary diagram in **R** by using `plot(x)`, with `x` an object of any compositional class (`acomp` or `rcomp`, see next section for details).

```
> xc = acomp(GeoChemSed, c("MgO","CaO","Na2O"))
> plot(xc)
```

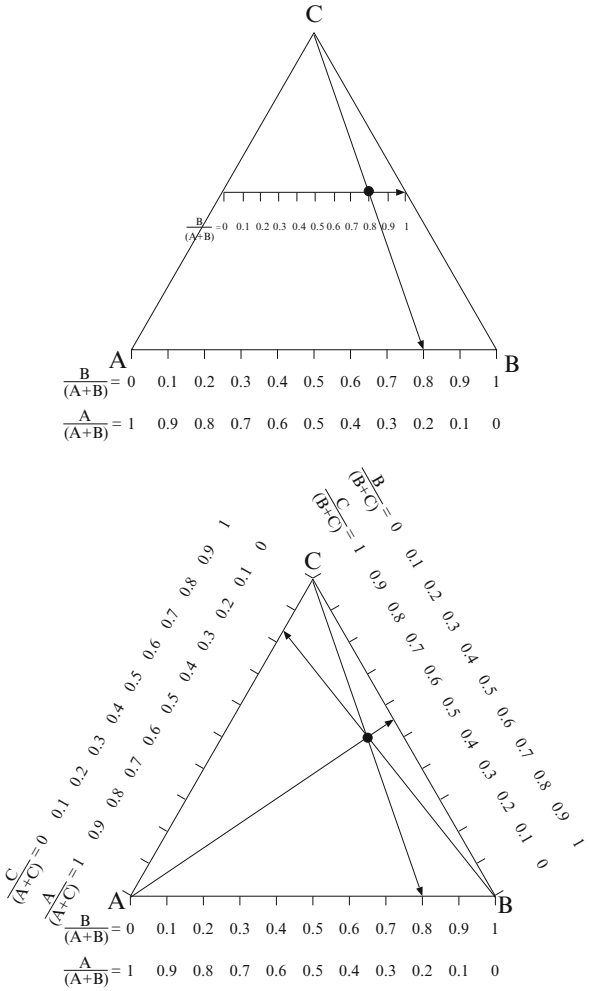


Fig. 2.4 The two part subcompositions can be read in two ways: (1) by projecting the composition to an edge of the triangle along a ray coming from the vertex of the part that should be ignored. Thus, all compositions along such rays have the same subcomposition, when the part where the ray is coming from is removed. (2) By drawing a scale between 0 and 1 on a segment parallel to the axis opposite to the part we want to remove, passing through our datum

2.3.3 Log-Ratio Scatterplots

Log-ratio scatterplots are just scatterplots of the log ratio of two parts against that of two other parts, though sometimes the denominators are the same. This representation is fully coherent with the description of compositions given in Sect. 2.1 but on the other side, does not capture any mass balance between the represented parts. In fact, it is "independent" of such mass balances: whenever we

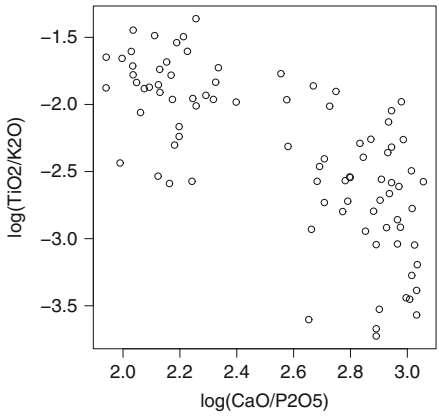


Fig. 2.5 Example of a log-ratio scatter plot obtained using the formula interface. This example was obtained with `plot(log(TiO2/K2O)~ log(CaO/P2O5),GeoChemSed)`. Recall that the formula syntax is `y ~ x`

cannot ensure that the studied system is not shrinking or growing, then log ratios of scatterplots display almost all the information we really have. For instance, if we are working with transported sediments, we do not know if any part can be dissolved by water and removed from the system; or when dealing with party vote shares between two different elections, we do not know if exactly the same people took part in the election. In fact, most of the datasets obtained in natural systems (meaning, not in the lab under strictly controlled situations) will be in such situations. Despite these issues, scatterplots of log ratios are seldom encountered in applications. To generate a log-ratio scatterplot, one can use the formula interface to `plot(formula, data)`, providing the `data` set where the variables involved will be extracted, and the desired `formula` describing the log ratios:

```
> plot(log(var1/var2)~log(var3/var4), dataset)
> plot(I(var1/var2)~I(var3/var4), dataset, log="xy")
```

Figure 2.5 shows an example obtained with the first syntax. These two syntax examples would only differ in the way axes are labeled, with the transformed values in the first case, with the original values of the ratios in the second case (using the command `I` is needed to tell **R** that the quotient inside it are not symbolic formulae, but actual ratios).

2.3.4 Bar Plots and Pie Charts

A *bar plot* is a classical representation where all parts may be simultaneously represented. In a bar plot, one represents the amount of each part in an individual as a bar within a set. Ideally, the bars are stacked, with heights adding to the total of the composition (1 or 100 %). Then, each individual is represented as one of such

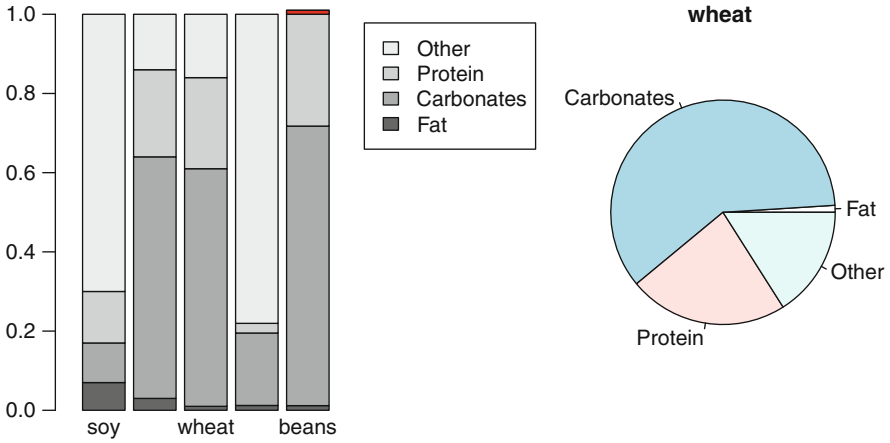


Fig. 2.6 Examples of bar plot (*left*) and pie plot (*right*). Note the presence of an extra segment on *top* of the bars for beans, showing the presence of a missing value

stacked bars, possibly ordered if that is possible (e.g., in time series). Stacked bars are provided in **R** by the command `barplot(x)`, when `x` is a compositional dataset. Individual compositions can also be displayed in the form of pie charts. Pie charts are produced by the `pie(x)` command, but now `x` must be a single composition (as only one pie diagram will be generated). Pie charts are not recommended for compositions of more than two parts, because the human eye is weak in the comparison of angles if they are not aligned (Bertin, 1967). Figure 2.6 is created by

```
> barplot( acomp(Amounts),xlim=c(0,11) )
> pie( acomp(Amounts["wheat",]), main="wheat")
```

2.4 Multivariate Scales

A fundamental property of each variable (or set of variables) in a dataset is its scale. The scale of a measurement describes which values are possible outcomes and which mathematical operations on these values are meaningful in the context of the application. Classical statistical practice distinguishes several scales for single variables, like (e.g., Stevens, 1946): categorical (several values are possible, but they are not comparable), ordinal (several values are possible, and they can be compared by ordering them), interval (any value in a subset of the real line is a possible outcome, absolute differences between them are meaningful), and ratio (any value in a subset of the *positive* real line is a possible outcome, relative differences between them matter). Other authors have criticized this list for being too limited: for instance, it does not include a separation between finite, infinite countable, and infinite uncountable scales (in any case, with interval or ratio differences).

Although a scale is above all a set of possible values together with some meaningful mathematical operations (and not a statistical model), each scale has

typical statistical models associated to them: e.g., the multinomial distribution is the most adequate and general model for categorical data, and the normal distribution appears as a central distribution for the real interval scale.

Compositional data add a further complexity to the scale issues as they are multivariate objects on themselves. In a composition, we cannot treat one isolated variable because its values are only meaningful in relation to the amounts of the other variables. The whole composition has thus to be considered as a unique object with some scale with joint properties, such as summing to 1 or 100 %, being positive, and obeying a ratio law of difference. A compositional scale is thus a new multivariate scale, not just the concatenation of several separate scales for each variable.

Compositional data has historically been treated in several ways according to different considerations on the underlying scale, and there was a lot of discussion on which is the *right* way of doing so. The "`compositions`" package provides several of these alternative approaches, with the aim of facilitating the comparison of results obtained following each of them. However, from our experience and the considerations presented in the first sections of this chapter, one of these scales should be the preferred one, as it is the one grounded in the mildest hypotheses: this is the Aitchison (1986) geometry for compositions, discussed in Sect. 2.5. Therefore, this book focuses on this scale, and the next section will introduce it in detail. But depending on our view of the problem, in some case, other scales might be more appropriate. They are available in the package and thus all presented and briefly discussed in this section for the sake of completeness. A comprehensive treatment of the multiple scale mechanisms in the package can be found in van den Boogaart and Tolosana-Delgado (2008).

To highlight the relevance of the choice of a scale, in "`compositions`", you assign it to the dataset *before any analysis*. This is done by a command like (e.g., `acomp` for **A**itchison **Comp**osition):

```
DataWithScale <- acomp(DataWithoutScale)
```

The scale of each compositional dataset is stored as its S3-class.[7] The package "`compositions`" then automatically treats the dataset according to the scale chosen. Each row of the dataset is considered as one observation in a compositional scale. The following multivariate scales are available, for both vectors of amounts and vectors of components (or actual compositions):

- "`rmult`"—**r**eal **mult**ivariate scale
- "`rplus`"—**r**eal interval restricted to the positive (**plus**) real orthant, $\mathbb{R}_+^D$
- "`aplus`"—**A**itchison (i.e., ratio) geometry in $\mathbb{R}_+^D$
- "`rcomp`"—**r**eal (i.e., interval) **comp**ositional scale
- "`acomp`"—**A**itchison (i.e., ratio) **comp**ositional scale
- "`ccomp`"—**c**ount **comp**ositional scale

[7] An S3-class is a concept in the object model of **R**. The class of an object determines how it is treated by generic functions, such as `plot` or `summary`. Use `help(class)` for further details on the concept in **R**.

Again, consider that the default choice of scale for compositional data should be the "acomp" Aitchison compositional scale.

2.4.1 Classical Multivariate Vectorial Data (rmult)

The real multivariate scale associated to the class "`rmult`" is the Cartesian product of D classical real scales, with values in a vector space $\mathbb{R}^D$. *This is neither a compositional nor an amount scale, as here negative values are fully meaningful.* The multivariate normal distribution is its central statistical model. Most well-known multivariate procedures were devised for this scale, such as cluster analysis, mean vector and variance–covariance matrix, principal component analysis, factor analysis, multivariate regression, and multivariate analysis of variance. Because it is the "conventional" scale, the package "`compositions`" only provides a limited functionality within it. It is mainly intended as a back-end for procedures based on treating the other scales through transformations. An example dataset displaying this scale might be the locations of earthquakes near Fiji Islands, data included in a standard **R** distribution:

```
> data(quakes)
> MultivariateData <- rmult(quakes[,c("lat","long","depth")])
```

2.4.2 Positive Data with Absolute Geometry (rplus)

Historically, the oldest approach of working with vectors of amounts is to see them as real multivariate data restricted to the positive orthant $\mathbb{R}^D_+$. Methods based on this approach are typically not scaling invariant, since each data is weighted proportional to its total amount (if nothing nastier happens). An associated distribution could be a truncated multivariate normal distribution. The way in which values of our sample can be changed is by simple addition or subtraction of some components, as long as the results remain positive. Analysis within the "`rplus`" framework is quite simple: apply the standard procedures from a multivariate scale to a dataset of positive values and hope for the best. It is thus easiest to analyze data in the "`rplus`" scale, as all statistical packages may be used for that end. However, several problems arise when interpreting results: for instance, in regression analysis, we will likely obtain negative predictions for an amount; if any procedure is based on assuming a normal distribution and our data have low means and high variances, truncation will be a source of problems; and last, but not least, if we are working with a vector of amounts with bounded sum (for instance, an incomplete composition), then correlation coefficients cannot be interpreted in the way we are used to, as they are spurious to an unknown extent (see Remark 2.1). In summary, the "`rplus`" scale is probably the most dangerous of all, because it is easily accessible in a conventional approach, but interpreting the results is a path full of pitfalls. An example from **R** illustration datasets might be the phosphor contents separated according to the

inorganic and organic part of the soil in an experiment analyzing the nutrient content of the resulting crop:

```
> data(phosphor)
> x <- rplus(phosphor,c("inorg","organic"))
```

2.4.3 Positive Data with Relative Geometry (aplus)

Stevens (1946) introduced the ratio scale to capture the notion that for a positive observational variable, only its relative variations might be meaningful. For several reasons, it is common to analyze such data after a log transformation. In this case, the reference distribution of statistical analysis becomes the lognormal distribution, and the natural way of describing how a variable evolves is through products/divisions. Later on, this idea has been extended to multivariate positive datasets (e.g. Pawlowsky-Glahn et al., 2003), giving rise to the "`aplus`" scale. The idea behind the approach is that the amount of each component can be individually analyzed in a relative scale, in the sense of ratio scale (Stevens, 1946), i.e., with component-wise product/division and distances computed based on the log-transformed data. Therefore, the natural central distribution in this case should be the multivariate lognormal. Though this is a common alternative to a full compositional approach, the analyst should be aware that most methods based on this approach are not scaling invariant.

An example might be the measures of morphology of iris flowers (length and width of petal and sepal), which are positive quantities with a relative geometry but where total size matters (since it discriminates between species):

```
> data(iris)
> x <- aplus(iris[,-1])
```

2.4.4 Compositional Data with Absolute Geometry (rcomp)

The most widely used way of treating compositional data is *as if they were* just multivariate real datasets that incidentally happened to be positive and sum to 1 or 100 %, but where these circumstances were irrelevant. Methods based on this approach are typically neither scaling nor perturbation invariant and definitively not subcompositionally coherent. A useful reference distribution for a D-part composition would be then a degenerate, $(D-1)$-dimensional multivariate normal distribution restricted to have all components positive and summing up to the constant. In this context, all changes between samples must be characterized by vectors of components summing up to *zero*, thus representing pure transference of "mass" between the parts.

Even though this naive approach was dominant, some special procedures were developed apart from the classical multivariate statistical toolbox, like the representation in ternary diagrams, and it was long warned that classical methods gave

meaningless results. Awareness of these problems stemmed from the Mathematical Geosciences community: problems linked to the spurious correlation (as it appeared already in the "`rplus`" scale) were reported for correlation and covariance (Butler, 1979; Chayes, 1960), principal components (Chayes and Trochimczyk, 1978), cluster analysis (Butler, 1978), autocorrelation (Pawlowsky-Glahn, 1984), etc. In fact, it was even shown that this kind of analysis does not obey the important principles of a meaningful statistical analysis of Sect. 2.2 (Aitchison, 1986). Nevertheless, the "`rcomp`" scale is still probably the most used of all scales, because of the same reasons that make the "`rplus`" scale dominant: the availability of software doing the job and the lack of awareness on the pitfalls of its interpretation.

But to be fair, we must also take into account that this scale honors a very important law of natural sciences: *mass balance* (Cortes, 2009; Shurtz, 2003), i.e., the fact that in a closed system (one where mass neither enters nor leaves the system), the total mass amount we have before and after any process must be preserved. This is most adequately captured by the vectors of zero sum describing change in this scale. Note nevertheless that the hypothesis of system closeness is a very strong one, that cannot come from the compositional datasets themselves, but from external, complementary information (Aitchison, 1986; Cortes, 2009). Furthermore, in the case that one wants to apply mass, matter, or volume preservation, the units of the dataset should be consistently masses, moles, or volumes. In this context, most changes of units would be nonsense.

2.4.5 Compositional Data with Aitchison Geometry (acomp)

In the early 1980s, Aitchison (1981, 1986) realized that in order to obtain meaningful results in a general framework, some basic principles had to be honored by any statistical analysis of compositional data. These principles have been stated in Sect. 2.2: scaling invariance, perturbation invariance, permutation invariance, and subcompositional coherence. He then showed that these principles imply a relative geometry, in which only the ratios of different components actually matter. In this context, a meaningful change between compositions has to be measured as a *perturbation* (as introduced in page 18, and more formally defined in Sect. 2.5), and the reference distribution becomes the additive-logistic-normal distribution, or normal distribution on the simplex (Chap. 3).

It was not until some years later that it was realized that the mathematical structure Aitchison had defined is indeed a vector space structure in its own right, isometrically equivalent to $\mathbb{R}^{D-1}$ (Barceló-Vidal, 2000; Billheimer et al., 2001; Pawlowsky-Glahn and Egozcue, 2001). A logical consequence of this equivalence is that we can convert any statistical problem involving compositions of D parts onto a classical multivariate problem involving real vectors of $(D-1)$ coordinates. This is what Pawlowsky-Glahn (2003) called "the principle of working in coordinates": whenever your scale is described by an Euclidean space structure, you automatically honor that scale if you statistically analyze the coordinates of the dataset with respect to any orthonormal basis (instead of treating the raw data). Section 2.5

presents in more detail this Euclidean space structure and the implications of the principle of working on coordinates for compositions. Note that the same principle of working on coordinates justifies analyzing log-transformed vectors of amounts if their scale is "`aplus`" (Pawlowsky-Glahn et al., 2003).

2.4.6 Count Compositions (ccomp)

Sometimes it is not possible to directly observe the "true" composition of a system but rather how many individuals belong to each of the groups in which this system is divided. We assume that the counts are proportional to the sizes of the groups. We call the result a *count composition*, because its elements are counts. A typical example are species compositions, such as how many insects of each taxon are found in an insect trap or how many people with blue, green, brown, and gray eyes are found in a classroom. The most apparent difference between both "`acomp`" or "`rcomp`" against "`ccomp`" scales is that the former are vectors of real numbers (thus can be called *real compositions*), whereas the latter are vectors of integers.

A typical model for count compositions is that conditioned to a fix total number of counts, the actual counts of each class are the outcomes of multinomial distribution with probabilities equal to the unknown, "true" composition, which we call the *underlying composition* (most probably, of "`acomp`" type).

Count compositions can be seen as an indirect observation of Aitchison compositions. However, this scale has some particular properties unlike the Aitchison compositional scale, as it has a dual nature. On one side, count compositions follow a relative scale, given that the actual count total (the number of insects in the trap, the number of people in the classroom) is irrelevant to understand the underlying process or distribution. But on another side, the statistical sampling error strongly depends on this total number of counts. Grossly simplifying, we can say that the total is irrelevant for the mean behavior of our system, but it matters when characterizing the variability of our observations and the uncertainty on our estimations. For example, in count compositions, a value of zero is easily possible as a random result if the corresponding component in the underlying composition is small (but not necessarily zero).

There are some practical problems with count compositions. For instance, plotting should take care to somehow represent how many times each of the possible count compositions was observed. There exist some competing models for count compositions (being log-linear models the most relevant). However, in our view, count compositions can be better fundamentally understood based on the properties of Aitchison compositions.

2.4.7 Practical Considerations on Scale Selection

This book is mainly devoted to the Aitchison composition scale ("`acomp`"). The other scales in the package are only discussed for background information.

However, the choice of a scale should not be driven by the goals of this book, but for an assessment of which is the meaningful scale for a problem at hand. This is necessarily a subjective decision that the analyst should thoroughly ground. Here are some guidelines.

One can select the right scale for the analysis based on the answers to three questions:

- *Are the components counts?*
  Many compositions are physically measured as counts, integer values, even if that is not relevant. For instance, many geochemical compositions are derived from counting how many atoms of a given type hit the sensor: but there are so many of them that we may safely consider that composition as a real composition. The same happens to the number of votes given in large electoral districts to the parties entering the parliament. On the other hand, when counting mineral or insect species, or the votes of all parties in a small district, the total is typically small, and we may be in a count composition case. Thus, the question must be stated more precisely: if this composition is formed by counts, *is their total so small that the discrete random selection error in the counts is important?* If the answer is *yes*, we are within a count composition problem.
- *Is the total mass in each observation an important aspect of the process under investigation?*
  If the answer is *yes*, the dataset should be treated with an amount scale, i.e., "`aplus`", "`rplus`" (if all variables are positive), or "`rmult`" (if they form a real multivariate dataset). If the answer is *no*, the dataset should be considered of a compositional scale, i.e., "`acomp`", "`rcomp`", or "`ccomp`".
- *Is the question of our focus only meaningful if the composition is expressed in a specific, common set of units? Does the composition contain all possible, potentially interesting components?*
  If the answer to *both* questions is *yes*, then the principles of scaling or perturbation invariance and subcompositional coherence are not necessary, and scales of an absolute type would be reasonable, e.g., "`rplus`" or "`rcomp`". This would imply that the structure under investigation is expected to be linear in real values of the components (e.g., a law of mass conservation). If the answer is *no* to any of these questions, then at least one of the principles is relevant, and we should only use relative scales ("`aplus`", "`acomp`", or "`ccomp`") to honor them.

Example 2.1 (Discussing the scale of a problem). The package "`compositions`" contains several example datasets for illustration that can be loaded with the command `data`. One of them contains chemical analyses of samples of water from rivers and streams of a medium-sized Mediterranean river basin from western Spain, including major (Na, Ca, Mg, K, Cl, SO_4, HCO_3) and minor ions (Ba, Sr) coming from the geological background, as well as other linked to urban sewage, industrial, or fertilizer pollution (total organic Carbon, H, NO_3, PO_4). Note that the data do not sum

up to a constant value. Now we can ask ourselves several questions about this dataset, leading to different choices of scales.

- An obvious question is which water samples are cleaner, or in other words, which carry a lower ionic concentration. It is obvious that in this case the total mass in each observation matters. It is also important which units do we choose to tackle with the problem, because we are going to obtain different answers when expressing our components in mol/L or in g/L. But, on the other hand, there are potentially thousands of pollutants that this dataset did not report. Thus, we should work with a scale of amounts, but the choice between "`rplus`" and "`aplus`" is open.
- A second question in this line would be which are the most important pollutants. In this case, the total amount still matters, but the units in which we are working (whether mol/L or g/L) do not: now it is more important to know the enrichment of each component on each sample in relation to a background value (for instance, the average); ammonia, for instance, being always present in small proportions, would not qualify as important for the first question but becomes a key element to answer the second question, because in polluted samples, it can be up to 150 times larger than in the background. Thus, we need a relative scale for amounts, the "`aplus`" one.
- Finally, we can also ask ourselves which are the geological factors conditioning the water chemistry. In this case, the total amount does not actually matter, as it is rather a function of dilution (how much water circulates) instead of the geology of our basin. Moreover, it is not good that our answer depends on the units we use. We should therefore treat the samples with an "`acomp`" scale. In fact, this sort of questions is usually answered by hydrogeochemists using the so-called Piper plot, a joint representation of four ternary diagrams representing the subcomposition of major ions in a "`rcomp`" scale.

In any case, once we have clear in mind our question and selected a scale, e.g., that of an Aitchison composition ("`acomp`"), you have to assign it to the dataset, as its S3-class:

```
> data(Hydrochem)
> dat <- acomp(Hydrochem[,6:19])
> class(dat)
```

```
[1] "acomp"
```

Further analysis of `dat` will then be automatically performed in a way consistent with the chosen compositional scale.

2.5 The Aitchison Simplex

2.5.1 The Simplex and the Closure Operation

A composition which elements sum up to a constant is called a *closed* composition, e.g., resulting from applying the closure operation (2.1). With "`compositions`", you can close a vector or a dataset using the command `clo`. The set of possible closed compositions

$$\mathbb{S}^D := \left\{ \mathbf{x} = (x_i)_{i=1,\dots,D} : x_i \geq 0, \sum_{i=1}^{D} x_i = 1 \right\}$$
$$= \left\{ \mathbf{x} = (x_1, \dots, x_D) : x_i \geq 0, \sum_{i=1}^{D} x_i = 1 \right\}$$

is called the D-part *simplex*. Throughout the book, notations $\mathbf{x}$, $(x_i)_i$, $(x_1, \dots, x_D)$ and $(x_i)_{i=1,\dots,D}$ will equivalently be used to denote a vector, in the two last cases showing how many components it has. The first two notations are reserved for cases where the number of components is obvious from the context or not important.

There is some discussion on whether the parts should be allowed to have a zero portion ($x_i = 0$) or not. For the scope of this book, zero values will be considered special cases, pretty similar to missing values (see Chap. 7).

The simplex of $D = 3$ parts is represented as a ternary diagram, where the three components are projected in barycentric coordinates (Sect. 2.3.2). In the same way, the simplex of $D = 4$ parts can be represented by a solid, regular tetrahedron, where each possible 3-part composition is represented on one side of the tetrahedron. To represent a given 3-part subcomposition of a 4-part dataset, we can project every 4-part datum inside the tetrahedron onto the desired side with projection lines passing through the opposite vertex, corresponding to the removed part. This is exactly the same as closing the subcomposition of the three parts kept. With help of that image, we can conceive higher-dimensional simplexes as hyper-tetrahedrons which projections onto 3- or 4-part sub-simplexes give ternary diagrams and tetrahedral diagrams.

2.5.2 Perturbation as Compositional Sum

The classical algebraic/geometric operations (addition/translation, product/scaling, scalar product/orthogonal projection, Euclidean distance) used to deal with conventional real vectors are neither subcompositionally coherent nor scaling invariant. As an alternative, Aitchison (1986) introduced a set of operations to replace these conventional ones in compositional geometry.

Perturbation plays the role of sum or translation and is a closed component-wise product of the compositions involved:

$$\mathbf{z} = \mathbf{x} \oplus \mathbf{y} = \mathscr{C}[x_1 \cdot y_1, \ldots, x_D \cdot y_D]. \tag{2.2}$$

With "compositions", one can perturb a pair of compositions in two ways: by using the command perturbe(x,y) or by *adding or subtracting* two vectors of class acomp, i.e.,

```
>  (x = acomp(c(1,2,3)))

[1] 0.1667 0.3333 0.5000
attr(,"class")
[1] acomp

>  (y = acomp(c(1,2,1)))

[1] 0.25 0.50 0.25
attr(,"class")
[1] acomp

>  (z = x+y)

[1] 0.125 0.500 0.375
attr(,"class")
[1] acomp
```

The neutral element of perturbation is the composition $\mathbb{1} = [1/D, \ldots, 1/D]$. It is an easy exercise to show that for any composition $\mathbf{x}$, the neutral elements fulfill $\mathbf{x} \oplus \mathbb{1} = \mathbf{x}$. The neutral element is thus playing the role of the zero vector. Any composition $\mathbf{x} = [x_1, \ldots, x_D]$ has its inverse composition: if one perturbs both, the result is always the neutral element. The inverse is $\ominus\mathbf{x} = \mathscr{C}[1/x_1, \ldots, 1/x_D]$ and it holds $\mathbf{x} \oplus \ominus\mathbf{x} = \mathbb{1}$. Perturbation by the opposite element plays the role of subtraction, hence the notation with $\ominus$. Perturbation with an inverse composition can as usually also be denoted with a binary $\ominus$ operator: $\mathbf{x} \ominus \mathbf{y} := \mathbf{x} \oplus \ominus\mathbf{y}$. In "compositions", inverse perturbation can be obtained subtracting two compositions, x-y or with perturbe(x,-y).

Example 2.2 (How and why do we perturb a whole dataset?). To perturb a whole compositional dataset by the same composition, we have to give both objects an acomp class and just ask for the dataset plus (or minus) the composition, for instance,

```
>  data(Hydrochem)
>  Xmas = Hydrochem[,c("Cl","HCO3","SO4")]
>  Xmas = acomp(Xmas)
>  mw = c(35.453, 61.017, 96.063)
>  mw = acomp(mw)
>  Xmol = Xmas-mw
```

`Xmas` contains the subcomposition Cl^-–HCO_3^-–$SO_4^=$ in mass proportions, and this is recasted to molar proportions in `Xmol` by dividing each component (*inverse perturbation*) with its molar weight (in `mw`). Most changes of compositional units may be expressed as a perturbation (between molar proportions, mass percentages, volumetric proportions, molality, ppm, etc.), as explained in Sect. 2.2.2. This operation is also applicable in the centering procedure of Sect. 4.1. Note that the two involved objects must have either the same number of rows (resulting in the perturbation of the first row of each dataset, the second of each dataset, etc.) or one of them must have only one row (resulting in all rows of the other object perturbed by this one).

In statistical analysis, it is often necessary to perturb or "sum up" all the compositions in the dataset. This is denoted by a big $\oplus$:

$$\bigoplus_{i=1}^{n} \mathbf{x}_i := \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_n$$

2.5.3 Powering as Compositional Scalar Multiplication

Powering or *power transformation* replaces the product of a vector by a scalar (geometrically, this is equivalent to scaling) and is defined as the closed powering of the components by a given scalar:

$$\mathbf{z} = \lambda \odot \mathbf{x} = \mathscr{C}[x_1^\lambda, \ldots, x_D^\lambda]. \tag{2.3}$$

Again, the package offers two ways to power a composition by a scalar: with the generic function `power(x,y)` or by *multiplying* a scalar by a vector of class `acomp`.

```
> 2*y
```

```
[1] 0.1667 0.6667 0.1667
attr(,"class")
[1] acomp
```

2.5.4 Compositional Scalar Product, Norm, and Distance

The *Aitchison scalar product* for compositions

$$\langle \mathbf{x}, \mathbf{y} \rangle_A = \frac{1}{D} \sum_{i>j}^{D} \ln \frac{x_i}{x_j} \ln \frac{y_i}{y_j} \tag{2.4}$$

provides a replacement for the conventional scalar product. The generic function `scalar(x,y)` has been added to "`compositions`" to compute scalar products: if `x` and `y` are `acomp` compositions, the result will be that of (2.4). Recall that the scalar product induces a series of secondary, useful operations and concepts.

- The norm of a vector, its *length*, is $\|\mathbf{x}\|_A = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_A}$. From an `acomp` vector `x`, its norm can be obtained with `norm(x)`. If `x` is a data matrix of class `acomp`, then the result of `norm(x)` will be a vector giving the norm of each row.
- The normalized vector of $\mathbf{x}$ (or its *direction*, or a unit-norm vector) is $\mathbf{v} = \|\mathbf{x}\|_A^{-1} \odot \mathbf{x}$. The generic function `normalize` provides this normalization, either for a single composition or for the rows of a data matrix.
- The angle between two compositions,

$$\alpha(\mathbf{x}, \mathbf{y}) = \cos^{-1} \frac{\langle \mathbf{x}, \mathbf{y} \rangle_A}{\|\mathbf{x}\|_A \cdot \|\mathbf{y}\|_A} = \cos^{-1} \left\langle \|\mathbf{x}\|_A^{-1} \odot \mathbf{x}, \|\mathbf{y}\|_A^{-1} \odot \mathbf{y} \right\rangle_A ,$$

  which allows to say that two compositions are *orthogonal* if their scalar product is zero, or their angle is 90°. This is not implemented in a single command, but you get it with

```
> acos(scalar(normalize(x),normalize(y)))
```

- The (orthogonal) projection of a composition onto another is the composition

$$P_{\mathbf{x}}(\mathbf{y}) = \frac{\langle \mathbf{y}, \mathbf{x} \rangle_A}{\langle \mathbf{x}, \mathbf{x} \rangle_A} \odot \mathbf{x},$$

  or in case of projecting onto a unit-norm composition, $P_{\mathbf{v}}(\mathbf{y}) = \langle \mathbf{y}, \mathbf{v} \rangle_A \odot \mathbf{v}$. This last case appears next when discussing coordinates with respect to orthonormal bases. A related concept is the (real) projection of a composition on a given direction, $\langle \mathbf{y}, \mathbf{v} \rangle_A$, which is a single real number.
- The *Aitchison distance* (Aitchison, 1986) between two compositions is

$$d_A(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} \ominus \mathbf{x}, \mathbf{y}\|_A = \sqrt{\frac{1}{D} \sum_{i=1}^{D} \sum_{j>i}^{D} \left(\ln \frac{x_i}{x_j} - \ln \frac{y_i}{y_j} \right)^2}. \tag{2.5}$$

  The distance between two compositions can be directly computed by `norm(x-y)`.

In "`compositions`", the generic function `dist` automatically computes the Aitchison distances between all rows of an `acomp` data matrix. The result is an object of class `dist`, suitable for further treatment (e.g., as input to hierarchical cluster analysis, see Sect. 6.2). Note that other compositional

distances, like Manhattan or Minkowski distances, are also available through the extra argument `method`:

```
> a = acomp(c(1,2,4))
> b = acomp(c(3,3,1))
> mycomp = rbind(a,b)
> mycomp = acomp(mycomp)
> dist(mycomp)

      a
b 1.813

> norm(a-b)

[1] 1.813

> dist(mycomp,method="manhattan")

      a
b 2.851

> sum(abs(clr(a-b)))

[1] 2.851
```

2.5.5 The Centered Log-Ratio Transformation (clr)

The set of compositions together with the operations perturbation $\oplus$, powering $\odot$ and Aitchison scalar product $(\cdot,\cdot)_A$ build a $(D-1)$-dimensional Euclidean space structure on the simplex. This means that we can translate virtually anything defined for real vectors to compositions, as an Euclidean space is always equivalent to the real space. This equivalence is achieved through an *isometry*, i.e., a transformation from the simplex to the real space that keeps angles and distances. The first isometric transformation we use is the *centered log-ratio transformation* (clr)

$$\mathrm{clr}(\mathbf{x}) = \left(\ln \frac{x_i}{g(\mathbf{x})} \right)_{i=1,\ldots,D} \quad \text{with} \quad g(\mathbf{x}) = \sqrt[D]{x_1 \cdot x_2 \cdots x_D}, \tag{2.6}$$

or in a compact way $\mathrm{clr}(\mathbf{x}) = \ln(\mathbf{x}/g(\mathbf{x}))$, where the log ratio of the vector is applied component-wise. The inverse clr transformation is straightforward: if $\mathbf{x}^* = \mathrm{clr}(\mathbf{x})$, then $\mathbf{x} = \mathscr{C}[\exp(\mathbf{x}^*)]$, where the exponential is applied by components. By its definition, the clr transformed components sum up to zero: in fact, the image of

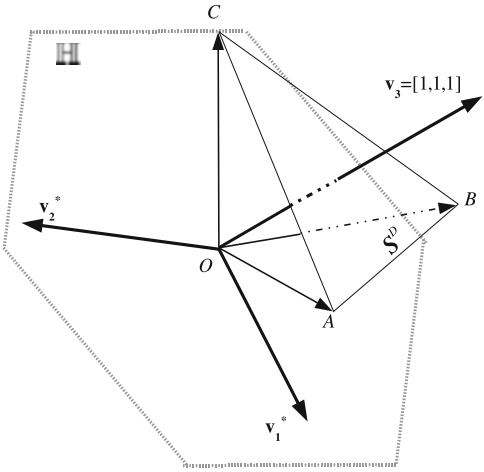


Fig. 2.7 Representation of a 3-part simplex (ABC, $\mathbb{S}^3$) and its associated clr-plane ($\mathbb{H}$), with an orthogonal basis $\{\mathbf{v}_1^*, \mathbf{v}_2^*, \mathbf{v}_3\}$ of $\mathbb{R}^3$. The reader can find illustrative to play with a 3-D representation of the clr-transformed data of a 3-part composition, like

```
> data(Hydrochem)
> idx = c(14,17,18)
> x = Hydrochem[,idx]
> x = acomp(x)
> plot3D(x, log=TRUE)
```

the clr is a hyperplane (and a vector subspace, denoted with $\mathbb{H}$, called the *clr-plane*) of the real space $\mathbb{H} \subset \mathbb{R}^D$ orthogonal to the vector $\mathbf{1} = [1, \ldots, 1]$, i.e., the bisector of the first orthant (Fig. 2.7). This may be a source of problems when doing statistical analyses, as e.g., the variance matrix of a clr-transformed composition is singular.

The commands `clr` and `clrInv` compute these two transformations: they admit either a vector (considered as a composition), or a matrix or data frame (where each row is then taken as a composition).

2.5.6 The Isometric Log-Ratio Transformation (ilr)

It is well known that there is only one $(D-1)$-dimensional Euclidean space typically called $\mathbb{R}^{D-1}$, up to an isometric mapping. Therefore, there exist an isometric linear mapping between the Aitchison simplex and $\mathbb{R}^{D-1}$. This mapping is called the *isometric log-ratio transformation* (ilr).

The isometry is constructed by representing the result in a basis of the $(D-1)$-dimensional image space $\mathbb{H}$ of the clr transformation. This is constructed by taking an *orthonormal basis* of $\mathbb{R}^D$ including the vector $\mathbf{v}_D = [1, \ldots, 1]$, i.e., some $(D-1)$ linearly independent vectors $\{\mathbf{v}_1^*, \ldots, \mathbf{v}_{D-1}^*\} \in \mathbb{H}$ (Fig. 2.7). Then the set of compositions defined as $\mathbf{v}_i = \mathrm{clr}^{-1}(\mathbf{v}_i^*)$ form a basis of $\mathbb{S}^D$. In computational terms, one can arrange the vectors $\{\mathbf{v}_j^*\}$ by columns in a $D \times (D-1)$-element matrix, denoted by $\mathbf{V}$, with the following properties:

- It is a quasi-orthonormal matrix, as

$$\mathbf{V}^t \cdot \mathbf{V} = \mathbf{I}_{D-1} \quad \text{and} \quad \mathbf{V} \cdot \mathbf{V}^t = \mathbf{I}_D - \frac{1}{D}\mathbf{1}_{D\times D}, \tag{2.7}$$

where $\mathbf{I}_D$ is the $D \times D$ identity matrix, and $\mathbf{1}_{D\times D}$ is a $D \times D$ matrix full of ones.

- Its columns sum up to zero, because they represent vectors of the clr-plane,

$$\mathbf{1} \cdot \mathbf{V} = 0. \tag{2.8}$$

Thanks to these properties, we can find simple expressions to pass between *coordinates* ξ and compositions $\mathbf{x}$:

$$\operatorname{clr}(\mathbf{x}) \cdot \mathbf{V}^t = \ln(\mathbf{x}) \cdot \mathbf{V}^t =: \operatorname{ilr}(\mathbf{x}) = \xi \tag{2.9}$$

$$\operatorname{ilr}(\mathbf{x}) \cdot \mathbf{V} = \operatorname{clr}(\mathbf{x}) \longrightarrow \mathbf{x} = \mathscr{C}\left[\exp\left(\xi \cdot \mathbf{V}\right)\right]. \tag{2.10}$$

Through these expressions, we also defined implicitly the so-called *isometric log-ratio transformation* (ilr): this is nothing else than a transformation that provides the coordinates of any composition with respect to a given orthonormal basis. There are as many ilr as orthonormal basis can be defined, thus as matrices $\mathbf{V}$ fulfilling (2.7) and (2.8). The one originally defined by Egozcue et al. (2003) was based on the (quite ugly) Helmert matrix

$$\mathbf{V} = \begin{pmatrix} \frac{D-1}{\sqrt{D(D-1)}} & 0 & \cdots & 0 & 0 \\ \frac{-1}{\sqrt{D(D-1)}} & \frac{D-2}{\sqrt{(D-1)(D-2)}} & \cdots & 0 & 0 \\ \frac{-1}{\sqrt{D(D-1)}} & \frac{-1}{\sqrt{(D-1)(D-2)}} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{-1}{\sqrt{D(D-1)}} & \frac{-1}{\sqrt{(D-1)(D-2)}} & \cdots & \frac{2}{\sqrt{6}} & 0 \\ \frac{-1}{\sqrt{D(D-1)}} & \frac{-1}{\sqrt{(D-1)(D-2)}} & \cdots & \frac{-1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{D(D-1)}} & \frac{-1}{\sqrt{(D-1)(D-2)}} & \cdots & \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{2}} \end{pmatrix}. \tag{2.11}$$

The ilr transformation induces an isometric identification of $\mathbb{R}^{D-1}$ and $\mathbb{S}^D$. For measure and probability theory purposes, this induces an own measure for the simplex, called the *Aitchison measure* on the simplex, denoted as $\lambda_{\mathbb{S}}$, and completely analogous to the Lebesgue-measure λ. This Aitchison measure is given by

$$\lambda_{\mathbb{S}}(A) = \lambda(\{\operatorname{ilr}(x) : x \in A\}).$$

The ilr transformation is available through the command `ilr(x)`. An optional argument `V` can be used to specify a basis matrix different from the default one. This is itself available through the function `ilrBase(x,z,D)`, where the arguments represent, respectively, the composition, its ilr transformation, and its number of parts (only one of them can be passed!). An alternative interface to some ilr transformations is provided by the functions `balance(x)` and `balanceBase(x)`, explained in Sect. 4.3. The commands `ilrInv` provide the inverse transformation. Also, if we want to pass from ilr to clr or vice versa, we can use functions `ilr2clr` and `clr2ilr`.

```
> a = c(1,2,4)
> (ac = acomp(a))

[1] 0.1429 0.2857 0.5714
attr(,"class")
[1] acomp

> ilr(ac)

       [,1]   [,2]
[1,] 0.4901 0.8489
attr(,"class")
[1] "rmult"

> clr2ilr(clr(ac))

[1] 0.4901 0.8489

> (Vd = ilrBase(x=a))

     [,1]    [,2]
1 -0.7071 -0.4082
2  0.7071 -0.4082
3  0.0000  0.8165

> clr(ac) %*% Vd

[1] 0.4901 0.8489
attr(,"class")
[1] "rmult"
```

2.5.7 The Additive Log-Ratio Transformation (alr)

Finally, it is worth mentioning that compositional data analysis in the original approach of Aitchison (1986) was based on the *additive log-ratio transformation* (*alr*), $\text{alr}(\mathbf{x}) = (\ln(x_1/x_D), \ldots, \ln(x_{D-1}/x_D)) = (\ln(x_i/x_D))_{i=1,\ldots,D-1}$. Though we will not use it in this book, the alr transformation is available with the command `alr(x)`.

Remark 2.2 (Why so many log-ratio transformations?). The answer is quite easy: because none is perfect. All three of them recast perturbation and

powering to classical sum and product,

$$\begin{aligned}\operatorname{clr}(\mathbf{x}\oplus\mathbf{y}) &= \operatorname{clr}(\mathbf{x}) + \operatorname{clr}(\mathbf{y}), \operatorname{ilr}(\mathbf{x}\oplus\mathbf{y}) = \operatorname{ilr}(\mathbf{x}) + \operatorname{ilr}(\mathbf{y}), \operatorname{alr}(\mathbf{x}\oplus\mathbf{y}) \\ &= \operatorname{alr}(\mathbf{x}) + \operatorname{alr}(\mathbf{y}) \\ \operatorname{clr}(\lambda\odot\mathbf{x}) &= \lambda\cdot\operatorname{clr}(\mathbf{x}), \operatorname{ilr}(\lambda\odot\mathbf{x}) = \lambda\cdot\operatorname{ilr}(\mathbf{x}), \operatorname{alr}(\lambda\odot\mathbf{x}) = \lambda\cdot\operatorname{alr}(\mathbf{x}).\end{aligned}$$

The first two, because they are isometric, also preserve the scalar product, but this does not happen with the alr transformation,

$$\langle\mathbf{x},\mathbf{y}\rangle_A = \operatorname{clr}(\mathbf{x})\cdot\operatorname{clr}^t(\mathbf{y}) = \operatorname{ilr}(\mathbf{x})\cdot\operatorname{ilr}^t(\mathbf{y}) \neq \operatorname{alr}(\mathbf{x})\cdot\operatorname{alr}^t(\mathbf{y})$$

Thus, alr should not be used in case that distances, angles, and shapes are involved, as it deforms them.

On the other side, we already mentioned that the clr yields singular covariance matrices, and this might be a source of problems if the statistical method used needs inverting it. One would need to use Moore–Penrose generalized inverses. As an advantage, the clr represents a one-by-one link between the original and the transformed parts, which *seems* to be helpful in interpretation.

This is exactly the strongest difficulty with the ilr-transformed values or any orthonormal coordinates: each coordinate might involve many parts (potentially all), which makes it virtually impossible to interpret them in general. However, the ilr is an isometry and its transformed values yield full-rank covariance matrices; thus, we can analyze ilr data without regard to inversion or geometric problems. The generic ilr transformation is thus a perfect black box: compute ilr coordinates, apply your method to the coordinates, and recast results to compositions with the inverse ilr. If interpretation of the coordinates is needed, one should look for preselected bases and preselected ilr transformations, as those presented in Sect. 4.3.

2.5.8 *Geometric Representation of Statistical Results*

The fact that the simplex is an Euclidean space has some implications on the way we apply, interpret and represent most linear statistics. The basic idea is summarized in the *principle of working on coordinates* (Pawlowsky-Glahn, 2003), stating that one should:

1. Compute the coordinates of the data with respect to an orthonormal basis (2.9).

2. Analyze these coordinates in a straightforward way with the desired method; no special development is needed, because the coordinates are real unbounded values.
3. Apply those results describing geometric objects to the orthonormal basis used, recasting them to compositions (2.10). Final results will not depend on the basis chosen.

Regarding the last step, most "geometric objects" obtained with linear statistics are points or vectors, lines/hyperplanes, and ellipses/ellipsoids.

Points (like averages and modes or other central tendency indicators, outliers, intersection points, etc.) are identified with their vectors of coordinates in the basis used. Thus, to represent a point as a composition, it suffices to apply its coordinates to the basis, i.e., compute the inverse ilr transformation (2.10). To draw points in a ternary diagram (or a series of ternary diagrams), one can use the function `plot(x)` where `x` is already a composition of class `acomp` (or a dataset of compositions). If points have to be added to an existing plot, the optional argument `add=TRUE` will do the job. This function admits the typical **R** set of accessory plotting arguments (to modify color, symbol, size, etc.).

Lines (like regression lines/surfaces, principal components, discriminant functions, and linear discriminant borders) are completely identified with a point $\boldsymbol{\alpha}$ and a vector $\boldsymbol{\beta}$ in the clr-plane. The parametric equation $\boldsymbol{\xi}(t) = \boldsymbol{\alpha} + t \cdot \boldsymbol{\beta}$ runs through every possible point of the line taking different values of the real parameter t. Applying this parametric equation to the basis in use, we obtain a compositional line,

$$\mathbf{x}(t) = \mathbf{a} \oplus t \odot \mathbf{b}, \tag{2.12}$$

with $\mathbf{a} = \text{ilr}^{-1}(\boldsymbol{\alpha})$ and $\mathbf{b} = \text{ilr}^{-1}(\boldsymbol{\beta})$. A P-dimensional plane needs P compositions of the second kind, i.e.,

$$\mathbf{x}(t_1, \ldots, t_P) = \mathbf{a} \oplus \bigoplus_{i=1}^{P} t_i \odot \mathbf{b}_i . \tag{2.13}$$

Note that a P-dimensional plane projected onto a subcomposition is still a P-dimensional plane. In particular, a line projected on a 3-part subcomposition is a line, obtained by selecting the involved parts in the vectors **a** and **b**. Straight lines can be added to existing plots by the generic function `straight(x,d)`, which arguments are respectively the point **a** and the vector **b** (as compositions!). Sometimes only a segment of the line is desired, and two functions might be helpful for that. Function `segments(x0,y)` draws a compositional line from `x0` to `y`, i.e., along `y-x0`; both *can* be compositional datasets, thus resulting in a set of segments joining `x0` and `y` row by row. Function `lines(x)` on the contrary *needs* a compositional dataset and draws a piece-wise (compositional) line through all the rows of `x`. Further graphical arguments typical of lines (width, color, style) can also be given to all these functions.

Ellipses and hyper ellipsoids (like confidence or probability regions or quadratic discriminant borders) are completely specified by giving a central point $\boldsymbol{\alpha}$ and a

symmetric positive definite matrix $\mathbf{T}$. An ellipse is the set of points $\boldsymbol{\xi}$ fulfilling

$$(\boldsymbol{\xi} - \boldsymbol{\alpha}) \cdot \mathbf{T} \cdot (\boldsymbol{\xi} - \boldsymbol{\alpha})^t = r^2, \tag{2.14}$$

namely, the set of points with norm r by the scalar product represented by the matrix $\mathbf{T}$. This can be extended to conics (and hyper quadrics) by dropping the condition of positive definiteness. Any hyper quadric projected onto a subcomposition is still a hyper quadric. In particular, in two dimensions (three parts), an ellipse is obtained with a positive definite matrix, a parabola with a semi-definite matrix, and an hyperbola with a non-definite matrix. All hyper quadrics show up as borders between subpopulations in quadratic discriminant analysis. But by far, their most typical use is to draw elliptic probability or confidence regions around the mean (see Example 4.2). In this case, we can better characterize the ellipses with the original variance matrix, i.e., $\mathbf{T} = \mathbf{S}^{-1}$. This is the approach implemented in the generic function `ellipses(mean, var, r)`, which arguments are a central composition (`mean`), a clr-variance matrix (`var`, described in Sect. 4.1), and the radius of the ellipse (`r`). Further graphical arguments for lines are also admitted.

Summary of graphical functions:

- `plot(x)` to draw ternary diagrams of the `acomp` composition `x`
- `plot(x, add=TRUE)` to add compositions to existing ternary diagrams
- `straight(x,d)` to draw lines on ternary diagrams along `d` passing through `x`, both `acomp` compositions
  - `segments(x0,y)` to add compositional segments from the rows of `x0` to the rows of `y`
  - `lines(x)` to add piece-wise compositional lines through `x` rows
- `ellipses(mean, var, r)` to draw an ellipse around the `acomp` composition `mean`, defined by a radius `r` and a clr-variance `var`

2.5.9 *Expectation and Variance in the Simplex*

In an Euclidean space, like the simplex, expectation and variance can be defined with regard to its geometry. Eaton (1983) presents the theory for any Euclidean space, and we will here present without proof the most important results in the Aitchison geometry of the simplex. The goal of this section is twofold. On the one hand, we want to see that the actual basis used to compute coordinates and apply statistics is absolutely irrelevant: back-transformed results are exactly the same whichever basis

was used. On the other hand, we will also find that variance matrices may be seen as *representations* of some objects on themselves, with a full meaning: this ensures us that whichever log-ratio transformation we use (clr, ilr or alr), expressions like (2.14) are fully correct as long as all vectors and matrices are obtained with regard to the same log-ratio transformation.

Take the Euclidean space structure of the simplex $(\mathbb{S}^D, \oplus, \odot, \langle\cdot,\cdot\rangle_A)$ and a random composition $\mathbf{X} \in \mathbb{S}^D$. Its *expectation within the simplex* is a fixed composition $\mathbf{m}$ such that the expected projection of any composition $\mathbf{v} \in \mathbb{S}^D$ onto $\mathbf{X}$ is perfectly captured by its projection onto $\mathbf{m}$,

$$\mathrm{E}_{\mathbb{S}}[\mathbf{X}] = \mathbf{m} \Leftrightarrow \text{for all } \mathbf{v} : \mathrm{E}\left[\langle \mathbf{v}, \mathbf{X}\rangle_A\right] = \langle \mathbf{v}, \mathbf{m}\rangle_A . \tag{2.15}$$

This says that knowing $\mathbf{m}$, we know the average behavior of $\mathbf{X}$ projected onto any direction of the simplex. This definition does not make use of any basis, so we can be sure that the concept itself is basis-independent. To compute it, however, we may take the vector $\mathbf{v}$ successively as each of the $(D-1)$ vectors of *any* ilr basis, thus obtaining the mean coordinates with respect to that basis,

$$\mathrm{ilr}\left(\mathrm{E}_{\mathbb{S}}[\mathbf{X}]\right) = \mathrm{E}[\mathrm{ilr}(\mathbf{X})] \in \mathbb{R}^{D-1}$$

This is the result we would obtain by applying the principle of working on coordinates. In other words, the definition given by (2.15) and the one implied by the principle of working on coordinates are equivalent, and both are in fact basis-independent.

For the variance, we may obtain equivalent results by defining it as an endomorphism. An endomorphism within the simplex is an application $\Sigma : \mathbb{S}^D \to \mathbb{S}^D$ that preserves the linearity of perturbation and powering, e.g., $\Sigma(\mathbf{x} \oplus \lambda \odot \mathbf{y}) = \Sigma(\mathbf{x}) \oplus \lambda \odot \Sigma(\mathbf{y})$. The *variance within the simplex* of a random composition $\mathbf{X} \in \mathbb{S}^D$ is an endomorphism Σ such that the expected product of the projections of any pair of compositions $\mathbf{u}, \mathbf{w} \in \mathbb{S}^D$ onto $\mathbf{X}$ (centered) is perfectly captured by their scalar product through Σ:

$$\mathrm{var}_{\mathbb{S}}[\mathbf{X}] = \Sigma \Rightarrow \mathrm{E}\left[\langle \mathbf{w}, \mathbf{X} \ominus \mathbf{m}\rangle_A \cdot \langle \mathbf{u}, \mathbf{X} \ominus \mathbf{m}\rangle_A\right] = \langle \mathbf{w}, \Sigma(\mathbf{u})\rangle_A = \langle \mathbf{u}, \Sigma(\mathbf{w})\rangle_A . \tag{2.16}$$

In words, by knowing Σ, we know the expected covariation of $\mathbf{X}$ projected onto any pair of directions of the simplex. As with expectation, the ilr-coordinate representation of Σ is a $(D-1) \times (D-1)$ matrix $\boldsymbol{\Sigma}_v = (\sigma^v_{ij})$ such that $\sigma^v_{ij} = \mathrm{cov}[\mathrm{ilr}_i(\mathbf{X}), \mathrm{ilr}_j(\mathbf{X})]$. And equivalently, its clr representation is a $D \times D$ matrix $\boldsymbol{\Sigma} = (\sigma_{ij})$ such that $\sigma^v_{ij} = \mathrm{cov}[\mathrm{clr}_i(\mathbf{X}), \mathrm{clr}_j(\mathbf{X})]$. Note that, whichever transformation we use,

$$\langle \mathbf{u}, \Sigma(\mathbf{w})\rangle_A = (\mathrm{ilr}(\mathbf{u}), \mathrm{ilr}(\Sigma(\mathbf{w}))) = \mathrm{ilr}(\mathbf{u}) \cdot \boldsymbol{\Sigma}_v \cdot \mathrm{ilr}^t(\mathbf{w}) = \mathrm{clr}(\mathbf{u}) \cdot \boldsymbol{\Sigma} \cdot \mathrm{clr}^t(\mathbf{w}).$$

Thus, all these matrices and vectors, expressed as compositions, in clr coefficients or in ilr coordinates, represent exactly the same objects. We can consistently change between them with the following expressions:

$$\begin{aligned} \text{ilr}(\mathbf{m}) = \text{clr}(\mathbf{m}) \cdot \mathbf{V} = \boldsymbol{\mu}_v \qquad & \mathbf{m} = \text{clr}^{-1}(\boldsymbol{\mu}_v \cdot \mathbf{V}^t) = \text{ilr}^{-1}(\boldsymbol{\mu}_v), \\ \boldsymbol{\Sigma}_v = \mathbf{V}^t \cdot \boldsymbol{\Sigma} \cdot \mathbf{V} \qquad & \boldsymbol{\Sigma} = \mathbf{V} \cdot \boldsymbol{\Sigma}_v \cdot \mathbf{V}^t. \end{aligned} \tag{2.17}$$

Functions `clrvar2ilr(xvar)` and `ilrvar2clr(zvar)` implement the variance transformations of (2.17), where `xvar` is the clr variance $\boldsymbol{\Sigma}$ and `zvar` is the ilr variance $\boldsymbol{\Sigma}_v$. To transform the mean vector, we may use the self-explaining functions `ilr`, `clr`, `ilrInv`, `clrInv`, `ilr2clr`, and `clr2ilr`.

References

Aitchison, J. (1981). Distributions on the simplex for the analysis of neutrality. In C. Taillie, G. P. Patil, & B. A. Baldessari (Eds.), *Statistical distributions in scientific work—Models, structures, and characterizations* (pp. 147–156). Dordrecht: D. Reidel Publishing Co., 455 pp.

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. London: Chapman & Hall (Reprinted in 2003 with additional material by The Blackburn Press), 416 pp.

Aitchison, J. (1997). The one-hour course in compositional data analysis or compositional data analysis is simple. In V. Pawlowsky-Glahn (Ed.), *Proceedings of IAMG'97—The third annual conference of the International Association for Mathematical Geology*, Volume I, II and addendum (pp. 3–35). Barcelona: International Center for Numerical Methods in Engineering (CIMNE), 1100 pp.

Aitchison, J., & Egozcue, J. J. (2005). Compositional data analysis: Where are we and where should we be heading? *Mathematical Geology, 37*(7), 829–850.

Barceló-Vidal, C. (2000). *Fundamentación matemática del análisis de datos composicionales*. Technical Report IMA 00-02-RR. Spain: Departament d'Informática i Matemática Aplicada, Universitat de Girona, 77 pp.

Bertin, J. (1967). *Semiology of graphics*. Madison: University of Wisconsin Press.

Billheimer, D., Guttorp, P., & Fagan, W. (2001). Statistical interpretation of species composition. *Journal of the American Statistical Association, 96*(456), 1205–1214.

Butler, J. C. (1978). Visual bias in R-mode dendrograms due to the effect of closure. *Mathematical Geology, 10*(2), 243–252.

Butler, J. C. (1979). The effects of closure on the moments of a distribution. *Mathematical Geology, 11*(1), 75–84.

Chayes, F. (1960). On correlation between variables of constant sum. *Journal of Geophysical Research, 65*(12), 4185–4193.

Chayes, F., & Trochimczyk, J. (1978). An effect of closure on the structure of principal components. *Mathematical Geology, 10*(4), 323–333.

Cortes, J. A. (2009). On the Harker variation diagrams; a comment on "the statistical analysis of compositional data. Where are we and where should we be heading?" by Aitchison and Egozcue (2005). *Mathematical Geosciences, 41*(7), 817–828.

Eaton, M. L. (1983). *Multivariate statistics. A vector space approach*. New York: Wiley.

Egozcue, J. J., Pawlowsky-Glahn, V., Mateu-Figueras, G., & Barceló-Vidal, C. (2003). Isometric logratio transformations for compositional data analysis. *Mathematical Geology, 35*(3), 279–300.

Pawlowsky-Glahn, V. (1984). On spurious spatial covariance between variables of constant sum. *Science de la Terre, Série Informatique, 21*, 107–113.

Pawlowsky-Glahn, V. (2003). Statistical modelling on coordinates. In S. Thió-Henestrosa & J. A. Martín-Fernández (Eds.), *Compositional data analysis workshop—CoDaWork'03, Proceedings*. Catalonia: Universitat de Girona. ISBN 84-8458-111-X, http://ima.udg.es/Activitats/CoDaWork03/.

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

Pawlowsky-Glahn, V., Egozcue, J. J., & Burger, H. (2003). An alternative model for the statistical analysis of bivariate positive measurements. In J. Cubitt (Ed.), *Proceedings of IAMG'03—The ninth annual conference of the International Association for Mathematical Geology* (p. 6). Portsmouth: University of Portsmouth.

Pearson, K. (1897). Mathematical contributions to the theory of evolution. On a form of spurious correlation which may arise when indices are used in the measurement of organs. *Proceedings of the Royal Society of London, LX*, 489–502.

Shurtz, R. F. (2003). Compositional geometry and mass conservation. *Mathematical Geology, 35*(8), 927–937.

Stevens, S. (1946). On the theory of scales of measurement. *Science, 103*, 677–680.

Tolosana-Delgado, R., & von Eynatten, H. (2010). Simplifying compositional multiple regression: Application to grain size controls on sediment geochemistry. *Computers and Geosciences, 36*, 577–589.

van den Boogaart, K. G., & Tolosana-Delgado, R. (2008). "compositions": A unified R package to analyze compositional data. *Computers and Geosciences, 34*(4), 320–338.

Chapter 3
Distributions for Random Compositions

Abstract Distribution models are descriptions of the randomness or uncertainty on a phenomenon. The most typical distribution models for compositions are the Dirichlet and the additive logistic normal (or normal on the simplex). For count compositions, the multinomial distribution and the double stochastic distribution are the reference models. The first two sections of this chapter summarize the definitions of these models, together with some other complementary probability models, as well as the available functions to work with them. The last section summarizes some properties and relations between the several distributions presented before. Tests for the various distribution models are discussed after presenting each model.

3.1 Continuous Distribution Models

3.1.1 The Normal Distribution on the Simplex

The normal distribution plays a capital role in statistics. It appears as the limit distribution of a phenomenon subject to additive "errors" or uncertainty sources, for instance, when computing the arithmetic average of any random variable. In case of a multidimensional phenomenon, one finds the multivariate normal distribution as the limit distribution of random translations (in length and direction). In the same way, the lognormal distribution appears when the "errors" are multiplicative. Compositions are multivariate variables by nature. And as the counterpart of translation in the simplex is perturbation, the normal distribution on the simplex appears as the limit distribution of a compositional phenomenon subject to random perturbations. In particular, it will appear as the limit distribution of the mean (4.1) of compositional dataset with finite compositional variance (see the chapter on descriptive statistics).

A random composition $\mathbf{X}$ has a *normal distribution on the simplex* (or *additive logistic normal*) with mean vector $\mathbf{m}$ and variance matrix $\boldsymbol{\Sigma}$, denoted as $\mathcal{N}_{\mathbb{S}}(\mathbf{m}, \boldsymbol{\Sigma})$, if projecting it onto *any* arbitrary direction of the simplex $\mathbf{u}$ with the Aitchison scalar

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_3, © Springer-Verlag Berlin Heidelberg 2013

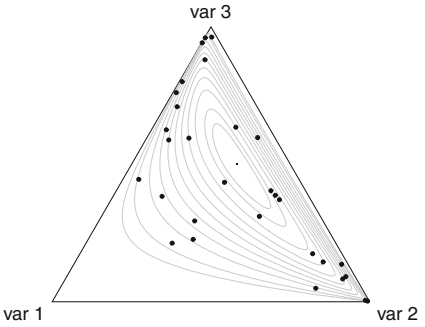


Fig. 3.1 Isodensity levels (*ellipses*) of an example normal distribution on the simplex, compared with a random sample (*dots*)

product (2.4) yields a random variable with a univariate normal distribution, of mean vector $\langle \mathbf{m}, \mathbf{u} \rangle_A$ and variance $\mathrm{clr}(\mathbf{u}) \cdot \boldsymbol{\Sigma} \cdot \mathrm{clr}^t(\mathbf{u})$. In particular, if we take a basis of the simplex (denoted by matrix $\mathbf{V}$, page 42), then the coordinates $\mathrm{ilr}(\mathbf{x})$ of the random composition have a multivariate normal distribution; i.e., their joint density with respect to the Aitchison measure $\lambda_{\mathbb{S}}$ is

$$f(\mathbf{x}; \boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v) = \frac{1}{\sqrt{(2\pi)^{(D-1)} \cdot |\boldsymbol{\Sigma}_v|}} \exp\left[-\frac{1}{2} (\mathrm{ilr}(\mathbf{x}) - \boldsymbol{\mu}_v) \cdot \boldsymbol{\Sigma}_v^{-1} \cdot (\mathrm{ilr}(\mathbf{x}) - \boldsymbol{\mu}_v)^t \right] \tag{3.1}$$

with $\boldsymbol{\mu}_v$ and $\boldsymbol{\Sigma}_v$, respectively, their mean vector and variance matrix. The two parameter sets are related through (2.17).

Function `dnorm.acomp(x, mean, var)` computes the density at a given point `x` of a normal distribution on the simplex with mean composition `mean` and clr-variance matrix `var`. An equivalent `rnorm.acomp(n, mean, var)` function generates `n` random compositional variates from this distribution. Figure 3.1 shows an example, obtained with

```
> opar <- par(mar=c(3,3,1,1))
> mymn = acomp( c(1,2,3) )
> myvr = ilrvar2clr( matrix(c(2,-1.5,-1.5,2),ncol=2) )
> plot(mymn,pch=".")
> for(p in c(0.5,1:9,9.5)/10){
+  r = sqrt(qchisq(p=p,df=2))
+  ellipses(mymn,myvr,r, col="grey")
+ }
> xr = rnorm.acomp(n=30, mean=mymn, var=myvr)
> plot(xr, add=TRUE, pch=19, cex=0.5)
> par(opar)
```

This chunk uses the property that the radius of a bivariate, independent, standard normal vector follows a χ^2_2 distribution. That is also true for the normal distribution on the simplex, just that now this is with respect to the Aitchison geometry.

The normal distribution on the simplex was defined in the present way by Pawlowsky-Glahn (2003) and studied in detail by Mateu-Figueras (2003). As probability distribution, it is equivalent to the additive-logistic-normal (ALN) distribution of Aitchison (1986), and the only difference between them is that the normal distribution in the simplex is expressed with respect to the Aitchison measure in the simplex and the additive logistic normal with respect to the Haar measure of the simplex as subset of $\mathbf{R}^D$. Thus, the ALN density is equal to (3.1) times the Jacobian of the transformation of a composition to ilr coordinates. Note that `dnorm.acomp` gives the normal density on the simplex and not the ALN density which is provided by *dlnorm.rcomp*. In this book, we use both *normal on the simplex* and *additive logistic normal* as synonyms.

Regarding other geometries (Sect. 2.4), functions `dnorm.rmult` and `dnorm.aplus`, respectively, provide the densities of the conventional multivariate normal and of the multivariate normal on the positive real line. This last one is given by

$$f(\mathbf{x};\mathbf{m},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \cdot |\boldsymbol{\Sigma}|}} \times \exp\Big[-\frac{1}{2}(\log(\mathbf{x}) - \log(\mathbf{m})) \cdot \boldsymbol{\Sigma}^{-1} \cdot (\log(\mathbf{x}) - \log(\mathbf{m}))^t\Big],$$

where $\mathbf{m}$ is the geometric mean and $\boldsymbol{\Sigma}$ the variance of the log-transformed variable. Its only difference with respect to the classical lognormal distribution is the absence of any Jacobian. The multivariate lognormal density can be obtained with function `dlnorm.rplus`. There are equivalent `rnorm.*` functions to generate random variates following all these distributions.

3.1.2 Testing for Compositional Normality

The test problem

H_0: X follows a normal distribution on the simplex.
H_1: X does not follow a normal distribution on the simplex

is very important, because the additive lognormal distribution is the typical reference for all linear models, and often one has to check that it satisfactorily fits to some data or some model residuals. Using the principle of working in coordinates, we can transform this into a test problem on multivariate normality of the ilr, clr, or alr transformed dataset. Unfortunately there are no well-established tests for multivariate normality available. We will thus discuss some alternatives, with their advantages and disadvantages.

3.1.2.1 Testing Marginals

It is well known that the projections of a normal dataset onto the directions of the original reference axes (the *marginals*) must necessarily be also normally

distributed. This is by no means a sufficient condition, but most practitioners do not go further and accept joint normality if the marginals are normally distributed. A sufficient condition would be to test that the projection onto *all* direction has a normal distribution.

Recall that for a compositional dataset $\mathbf{X}$, marginals have no meaning, because we must work at least with two parts. In compositional data, we must always test projections or, alternatively, subcompositions. Two strategies have been suggested so far to test joint compositional normality.

The first option, implemented in the function `qqnorm.acomp(y,alpha)`, checks that all 2-part subcompositions have a normal distribution. Though (again) this is not sufficient to ensure joint normality, it offers a good approximation: we indeed test $D(D-1)/2$ directions, the number of differences between two parts, quite more than the naive $(D-1)$ marginals. The function itself generates a matrix of QQ plots: in each cell, the series of empirical quantiles of the log ratio of the two parts given by row and column is plotted against the equivalent expected quantiles for a standard normal distribution. A good approximation to normality is displayed when the resulting points show a linear pattern. If we give a value between 0 and 1 for the accessory parameter `alpha` (`NULL` by default), then the function computes a Shapiro–Wilk test of normality of each log ratio at a significance level `alpha` *divided by* $D(D-1)/2$: log ratios displaying a significant departure from normality at that corrected alpha level are marked by a red exclamation mark.

3.1.2.2 A Test Based on SVD

The second option, put forward by Aitchison et al. (2004), proposes to do three kinds of tests on the scores of a singular value decomposition (SVD), thus to work (at most) on a set of $(D-1)$ orthogonal, representative directions. One first applies the SVD to the *centered, clr-transformed* compositional dataset (see Sect. 6.1.1) and chooses how many singular values will be considered different than zero. That number is the rank R of the dataset and gives the maximum order at which the following tests will be conducted:

1. Each column of the scores matrix $\mathbf{U} = [u_{ni}]$ should show a standard normal distribution:

$$\{u_{ni},\, n = 1, \ldots, N\} \sim \mathcal{N}(0,1), \quad i = 1, \ldots, R;$$

   the test can be actually computed with the function `shapiro.test` for a Shapiro–Wilk test of normality, or with the Kolmogorov–Smirnov test function `ks.test`, describing the null-hypothesis distribution with the optional arguments `y="pnorm", mean=0, sd=1`.
2. As the scores columns should be uncorrelated, the angle between each pair of columns should show a uniform distribution on the circle:

$$\{\theta_{n,(ij)} = \tan^{-1}(u_{ni}/u_{nj}),\ n = 1, \ldots, N\} \sim \mathscr{U}(-pi, \pi),$$
$$i, j = 1, \ldots, R, \quad i \neq j;$$

the test is computed with the Kolmogorov–Smirnov test function `ks.test`, giving the extra arguments `y="punif", min=-pi, max=pi` to describe the distribution of the null hypothesis.

3. The squared norms of each row of the scores matrix follow a χ^2 distribution:

$$\{|\mathbf{u}_n|^2,\ n = 1, \ldots, N\} = \sum_{i=1}^{R} u_{ni}^2 \sim \chi^2(R);$$

the test will be provided by a call to `ks.test`, with null hypothesis specified via the arguments `y="pchisq"` and `df` equal to the chosen rank, R.

The total number of tests to conduct is $1 + R(R+1)/2$. Thus, the desired global significance should be divided by that number to get the significance level to use in each test. This second testing option is not (yet) encapsulated in any **R** function, and it should be computed "manually" step by step. Note that most of the procedure can actually be applied to any multivariate dataset, as the compositional nature of the data plays a role only when computing the SVD.

3.1.2.3 Complete Multivariate Test

The third option to test compositional normality is based on the energy test of Rizzo and Szekely (2008), implemented in the package "`energy`". This test must be applied to an ilr transformation (*any*) of the data:

```
> mvnorm.etest(ilr(sa.lognormals5))
> mvnorm.etest(ilr(sa.dirichlet5))
```

The test is based on the test statistic (Szekely and Rizzo, 2005):

$$E = n\left(\frac{2}{n}\sum_{1}^{n} E[\|x_i - Z\|] - E[\|Z - Z'\|] - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x_j\|\right) \tag{3.2}$$

where $y_1, \ldots, y_n$ is the standardized sample and Z, Z' standard normal variates. The test is thus not very sensitive to local deviations form normality. An interested user can check the adequate coverage of the test by simulation, e.g.,

```
> hist(replicate(1000,
+  mvnorm.etest(cbind(rnorm(20),rnorm(20),rnorm(20)))$p.value))
```

will generate a quite uniform histogram of the p-values of 1,000 datasets of 20 simulated 3-variate normal vectors. A similar test with an adapted test statistic

is built into "compositions" with command acompNormalGOF.test, as will be illustrated in the following example.

Example 3.1 (Testing for compositional normality). The global test implemented in "compositions" yields one single p-value that evaluates global normality, but in case of deviations from additive logistic normality, it cannot be used to explore where these deviations occur. In this case, QQ plots and the set of SVD tests can help. Let us compare what happens between three simulated datasets: one clearly ALN-distributed (a closed sa.lognormals5), one clearly not ALN-distributed (a closed sa.dirichlet5), and one mixed, formed taking the first three columns of sa.lognormals5 and the last two of sa.dirichlet5:

```
> sa.mixed = cbind(sa.lognormals5[,1:3], sa.dirichlet5[,4:5])
> sa.mixed = acomp( sa.mixed )
```

Applying the global test we find:

```
> acompNormalGOF.test(sa.lognormals5)

    Energy test of multivariate normality: estimated
    parameters

data:  x, sample size 60, dimension 4, replicates 999
E-statistic = 1.035, p-value = 0.2763

> acompNormalGOF.test(sa.dirichlet5)

    Energy test of multivariate normality: estimated
    parameters

data:  x, sample size 60, dimension 4, replicates 999
E-statistic = 2.4, p-value < 2.2e-16

> acompNormalGOF.test(sa.mixed)

    Energy test of multivariate normality: estimated
    parameters

data:  x, sample size 60, dimension 4, replicates 999
E-statistic = 1.941, p-value < 2.2e-16
```

which tells us that acomp(sa.lognormals5) is globally ALN-distributed, while the other two are not. We want to explore why acomp(sa.mixed) is not, thus proceeding to a singular value decomposition of the clr-transformed data and the application of Shapiro–Wilk tests to each column of the scores matrix. First, we *center* and *clr-transform* the data

```
> u = svd( scale(clr(sa.mixed), center=TRUE, scale=FALSE) )$u
```

and then define and apply the test:

```
> mytest = function(...) shapiro.test(...)$p.value
> apply(u, 2, mytest)
```

```
[1] 1.274e-01 3.513e-04 5.425e-01 2.282e-02 8.207e-07
```

Results suggest that non-normality is concentrated in 2 dimensions of the four of the problem. The complete set of tests would be obtained with the angle tests,

```
> mytest = function(i,j){
+   ks.test(atan2(u[,i], u[,j]), y="punif", min=-pi,
                                        max=pi)$p.value
+   }
> autoload("combn",package="combinat")
> (pairstocompare = combn(1:5,2))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    1    1    2    2    2    3    3     4
[2,]    2    3    4    5    3    4    5    4    5     5
```

```
> sapply(pairstocompare, mytest)
```

```
 [1] 1.214e-01 9.522e-06 1.214e-01 5.997e-02 1.214e-01 1.055e-02
 [7] 1.214e-01 0.000e+00 9.522e-06 5.997e-02 9.522e-06 1.055e-02
[13] 9.522e-06 0.000e+00 5.997e-02 1.055e-02 5.997e-02 0.000e+00
[19] 1.055e-02 0.000e+00
```

and the radius test,

```
> radius = sqrt(rowSums(u^2))
> ks.test(radius, y="pchisq", df=4)$p.value
```

```
[1] 3.331e-16
```

A QQ-ALN plot (Fig. 3.2) shows that non-normality is concentrated in the ratios involving the last two components, as expected by the way we constructed this simulated dataset:

```
> opar <- par(oma=c(1,1,1,1))
> qqnorm(sa.mixed, alpha=0.05)
> par(opar)
```

The reader is invited to compare this figure with the equivalent diagrams for the other two datasets:

```
> qqnorm(acomp(sa.lognormals5), alpha=0.05)
> qqnorm(acomp(sa.dirichlet5), alpha=0.05)
```

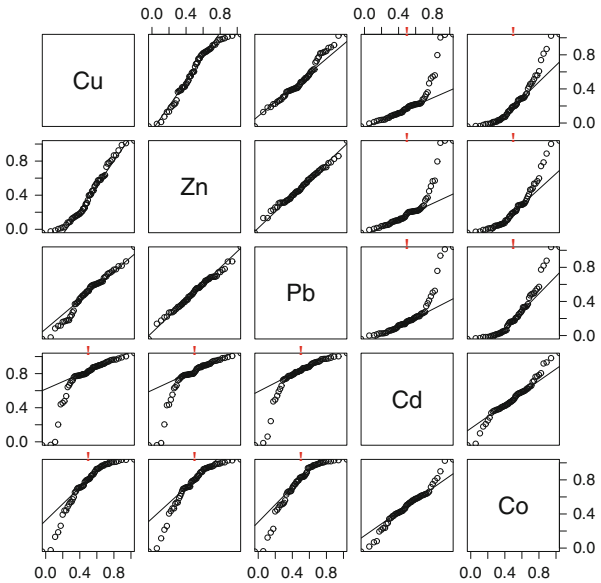


Fig. 3.2 QQ normality plots of all pairwise log ratios. This is the result of `qqnorm(x)` with an `acomp` object. The accessory argument `alpha=0.01` (or any other significance level) marks which log ratios deviate from normality at that significance

3.1.3 The Dirichlet Distribution

The *Dirichlet distribution* is the distribution of a composition obtained as the closure of a vector of D independent, equally scaled (i.e., with the same λ parameter) gamma-distributed variables. Thus, if $x_i \sim \Gamma(\lambda, \alpha_i)$, then $\mathbf{z} = \mathscr{C}[\mathbf{x}] \sim \mathscr{D}i(\boldsymbol{\alpha})$, where $\mathbf{x} = [x_i]$ and $\boldsymbol{\alpha} = [\alpha_i]$ are vectors of D positive components. The Dirichlet density is classically given as

$$f(\mathbf{z};\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_D)} \frac{z_1^{\alpha_1}\cdots z_D^{\alpha_D}}{z_1\cdots z_D},$$

where $\alpha_0 = \alpha_1 + \cdots + \alpha_D$ and $\Gamma(\cdot)$ is the gamma function. The package "`compositions`" offers the possibility to generate variates from this distribution, by means of the function `rDirichlet.acomp(n, alpha)`, where `n` is the desired number of random compositions and `alpha` is the vector $\boldsymbol{\alpha}$ of positive parameters. The output of this function has an `acomp` class. To compute the Dirichlet density, one can use the function `ddirichlet(x, alpha)`, available in other packages like "`MCMCpack`" and "`gtools`".

Arithmetic mean and variance of the Dirichlet distribution are

$$E[\mathbf{z}] = \mathscr{C}[\boldsymbol{\alpha}] \quad \text{and} \quad \mathrm{var}_{\mathbb{R}^D}(z) = \frac{1}{\alpha_0 + 1}(\mathrm{diag}(\bar{\mathbf{z}}) - \bar{\mathbf{z}}^t \cdot \bar{\mathbf{z}}).$$

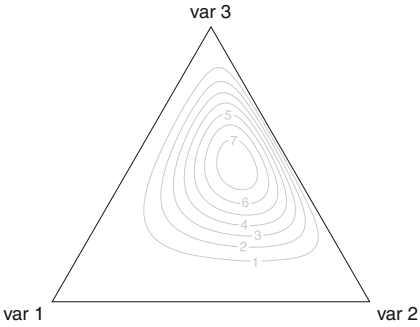


Fig. 3.3 Isodensity levels of an example Dirichlet distribution

The compositional moments (geometric center and clr variance) are

$$E_A[\mathbf{z}] = \text{clr}^{-1}[\psi(\boldsymbol{\alpha})] \quad \text{and} \quad \text{var}_A(z) = \mathbf{H} \cdot \text{diag}(\psi'(\boldsymbol{\alpha})) \cdot \mathbf{H},$$

being $\mathbf{H} = \mathbf{I}_D - \frac{1}{D}\mathbf{1}_{D\times D}$ and the functions $\psi(\cdot)$ and $\psi'(\cdot)$ the `digamma` and `trigamma` functions, respectively, applied to $\boldsymbol{\alpha}$ component-wise. Note also that diag(·) is meant to be a diagonal matrix which diagonal elements are provided by the argument of the function. Proofs of these statements can be found in Aitchison (1986) and Mateu-Figueras and Pawlowsky-Glahn (2005).

Note that both variance matrices (the raw and the clr-transformed) are singular and derived from an "uncorrelated correlation" structure (a diagonal matrix). This is inherited from the fact that the original gamma-distributed values were independent: because of this parental independence, the Dirichlet distribution is extremely ill-posed to model codependence between components.

Example 3.2 (Plotting a scalar function of a composition). Isodensity levels (Fig. 3.3) of the Dirichlet distribution are not elliptic, either in the classical geometry or in the Aitchison geometry of the simplex. This makes getting a plot of its density quite tricky:

```
> library("gtools")
> opar <- par(mar=c(3,3,1,1))
> myalpha = c(1,2,3) + 1
> plot(acomp(myalpha),pch="")
> aux = seq(from=0, to=1, by=0.01)
> myx = expand.grid(x=aux,y=aux)
> c60 = cos(pi/3)
> s60 = sin(pi/3)
```

```
> myfun = function(x){
+  y = c(x[1]-x[2]*c60/s60, x[2]/s60)
+  y = c(1-sum(y),y)
+  dd = ifelse(any(y<0),NA, ddirichlet(y, alpha=myalpha ) )
+  return(dd)
+ }
> dx = apply(myx,1,myfun)
> dim(dx) = c(101,101)
> contour(dx,asp=1,add=TRUE,col="grey")
> par(opar)
```

Here we generated a square grid of pairs of values between 0 and 1 (by 0.01), stored in `myx`. Then, we took each of these values as the Cartesian coordinates behind a ternary diagram, recasted those to barycentric coordinates (thus, to compositions) and, then, for those points inside the ternary diagram computed the Dirichlet density. The trick here is to build an ad hoc function (`myfun`) doing the change of coordinates and density computation for one point and then apply this function to each row of the grid (through the command `apply`). Note that the package "`gtools`" (providing access to the Dirichlet density) can be replaced by the package "`MCMCpack`". Command `contour` is a graphical function of **R** base distribution. This chunk can be recycled for plotting other densities or scalar functions of a composition, by changing the `ddirichlet` call to the desired function.

To understand the Dirichlet distribution, we can add to the plot a random sample of the same Dirichlet

```
> xr = rDirichlet.acomp(n=30, alpha=myalpha)
> plot(xr,pch=19,cex=0.5,add=TRUE)
```

or its arithmetic and compositional means,

```
> Dstats = acomp(rbind(clo(myalpha), clrInv(digamma(myalpha))))
> plot(Dstats,pch=19,cex=0.5,add=TRUE)
```

This is left to the reader. Try also with different values for `myalpha`, in particular removing the $+1$ of the third line of code.

3.1.3.1 Testing for Dirichlet Distributions

An experimental goodness of fit test for the Dirichlet distribution is available by the `acompDirichletGOF.test` command:

```
> acompDirichletGOF.test(acomp(sa.uniform5),R=10000)

    Compositional test for Dirichlet distribution

data:  acomp(sa.uniform5)
E-statistic = 0.2152, p-value = 0.9823
```

The compositional goodness of fit testing problem is essentially a multivariate goodness of fit test. However, **R** does not provide such standardized multivariate goodness of fit tests. Thus, "`compositions`" includes an omnibus multivariate goodness of fit test, which is base on the two-sample test `eqdist.test` for the same distributions from the energy package (Rizzo and Szekely, 2008). `acompDirichletGOF` essentially fits a Dirichlet distribution, simulates a large sample, and checks for equal distribution.

3.1.4 The Aitchison Distribution

Aitchison (1985) presented a family of distributions which generalize both the Dirichlet distribution and the additive logistic normal (or the normal distribution on the simplex). A random composition $\mathbf{X}$ has an *Aitchison distribution* with D component location vector $\boldsymbol{\theta}$ and spread $(D-1)$-quadratic matrix $\boldsymbol{\beta}_v$ if its log-density (with respect to the classical Lebesgue measure) can be expressed as

$$\log f(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\beta}_v) = -\kappa(\boldsymbol{\theta},\boldsymbol{\beta}_v) + (\boldsymbol{\theta}-\mathbf{1})\cdot\log(\mathbf{x})^t + \mathrm{ilr}(\mathbf{x})\cdot\boldsymbol{\beta}_v\cdot\mathrm{ilr}(\mathbf{x})^t, \qquad (3.3)$$

where $\kappa(\boldsymbol{\theta},\boldsymbol{\beta}_v)$ is a constant (no analytical form) ensuring that the density will integrate to one. This is only possible if:

1. $\boldsymbol{\beta}_v$ is a symmetric negative definite quadratic form and $\sum_{i=1}^{D}\theta_i \geq 0$.
2. $\boldsymbol{\beta}_v$ is symmetric but not positive definite, and $\theta_i > 0$ for all $i = 1,\ldots,D$.

In the present versions of "`compositions`", functions dealing with the Aitchison distribution work only with parameters satisfying the first condition. In this case, a second more interpretable parameterization can be used. By taking $\boldsymbol{\Sigma}_v$ a positive definite $(D-1)$-quadratic matrix (a quasi-variance), $\mathbf{m}$ a D-part composition (a quasi-mean), and α a positive scalar, the density of the Aitchison distribution can then be expressed as

$$f(\mathbf{x};\alpha,\mathbf{m},\boldsymbol{\Sigma}_v) = \exp\left[-\kappa(\alpha,\mathbf{m},\boldsymbol{\Sigma}_v)\right]\times\exp\left[(\alpha-1)\cdot\log(\mathbf{x})\cdot\mathbf{1}^t\right] \times\exp\left[-\frac{1}{2}\mathrm{ilr}\left(\mathbf{x}\ominus\mathbf{m}\right)\cdot\boldsymbol{\Sigma}_v^{-1}\cdot\mathrm{ilr}^t\left(\mathbf{x}\ominus\mathbf{m}\right)\right], \qquad (3.4)$$

which is proportional to the product of a symmetric Dirichlet distribution density (only depending on α) and a normal distribution density on the simplex (defined by $\mathbf{m}$ and $\boldsymbol{\Sigma}_v$). Due to this double parameterization, the Aitchison distribution can be either denoted as $\mathscr{A}(\boldsymbol{\beta},\boldsymbol{\theta})$ or as $\mathscr{A}(\alpha,\mathbf{m},\boldsymbol{\Sigma})$. "`compositions`" allows

both sets of parameters to be given to the functions `dAitchison(x,...)` and `rAitchison(n,...)`, which respectively compute the density at `x` and generate `n` random variates. Either `beta` or `sigma` must be given to indicate the spread of the distribution; both can be specified as clr or ilr matrices, related through (2.17). And for the location vector, one must either provide `theta` or else `alpha` and `m` (an `acomp` composition).

The two sets of parameters are related through the following equations (where $\mathbf{V}$ is the ilr matrix in use),

$$\begin{aligned}
\boldsymbol{\Sigma}_v &= -\frac{1}{2}\boldsymbol{\beta}_v^{-1}, \qquad & \boldsymbol{\beta}_v &= -\frac{1}{2}\boldsymbol{\Sigma}_v^{-1}, \\
\alpha &= \sum_{i=1}^{D} \theta_i / D, & & \\
\mathrm{ilr}(\mathbf{m}) &= \boldsymbol{\Sigma}_v \cdot \mathbf{V}^t \cdot \boldsymbol{\theta}, \qquad & \boldsymbol{\theta} &= \mathrm{clr}(\mathbf{m}) \cdot \mathbf{V} \cdot \boldsymbol{\Sigma}_v^{-1} + \alpha \mathbf{1}.
\end{aligned}$$

Note $\mathbf{m}$ and $\boldsymbol{\Sigma}_v$ are *not* the mean composition and variance matrix of the Aitchison distribution, but just approximate values. The difference between these quasi-moments and the actual moments depends on α: this parameter is interpreted as a measure of departure of the Aitchison distribution from the normal on the simplex. As no analytical expression of these moments exists for any $\alpha \neq 0$, the package provides them in a numerically computed accessory function, called `AitchisonDistributionIntegrals`, also admitting both sets of parameters. The interested reader is advised to read the help page of this function and the original definitions of the distribution (Aitchison, 1985, 1986).

3.2 Models for Count Compositions

3.2.1 The Multinomial Distribution

The conditional *multinomial distribution* is the most simple model for count compositions. If a vector $\mathbf{X}$ of D natural numbers summing up to N has a multinomial distribution of parameters $\mathbf{p}$ and N, then its density is

$$f(\mathbf{x}; \mathbf{p}, N) = N! \prod_{i=1}^{D} \frac{p_i^{x_i}}{x_i!}.$$

This is the distribution of N trials of an experiment with D possible outcomes A_i, which probabilities are known p_i. In this context, the components of $\mathbf{x}$ represent the number of times the outcome A_i was observed, out of the total N. The key idea of the multinomial distribution is that the vector of probabilities $\mathbf{p} = [p_1, \ldots, p_D]$

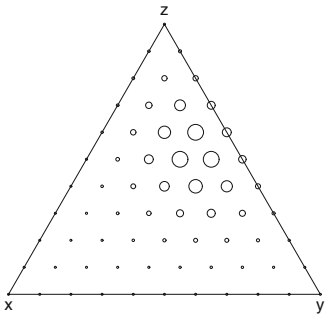


Fig. 3.4 An example of a multinomial distribution: note that the size of the triangle is this time 10 units

(incidentally, a composition of total sum 1) is the same for all trials and the outcome of each trial is *independent* of the outcome of the preceding ones.

The expectation vector and variance matrix of this distribution are, respectively, $N \cdot \mathbf{p}$ and $N(\text{diag}(\mathbf{p}) - \mathbf{p}^t \cdot \mathbf{p})$. Note that the rows and columns of the variance matrix sum up to zero, because the vector $\mathbf{x}$ has a constant sum equal to N.

The multinomial distribution is available in the basic **R** distribution, with functions `rmultinom(n, size, prob)` and `dmultinom(x, size, prob)`. The first function generates `n` samples of a multinomial distribution with parameters: N given by `size` and probability vector $\mathbf{p}$ specified in `prob`. The second function provides the elementary probability of this multinomial distribution for `x`.

Another useful function is included in package "`combinat`", devoted to combinatorics: command `xsimplex(p,n)` computes the so-called (p, n)-*simplex*, i.e., all the ways in which a natural number `n` can be expressed as the sum of `p` natural numbers. This is the set of support points of the multinomial density. Note that, in this last function, the argument `n` corresponds to the number of trials N, given in argument `size` in the first two functions. Also, `p` is the length of the probability vector. Figure 3.4 shows an example, obtained with

```
> library("combinat")
> opar <- par(mar=c(3,3,1,1))
> mysx = t( xsimplex(p=3,n=10) )
> dx = apply(mysx, 1, function(x){
+       dmultinom(x, size=10, prob=clo(mymn))
+     })
> plot(rcomp(mysx), cex=0.25+dx/max(dx)*1.75)
> par(opar)
```

A common error when working with counts is to take $\mathbf{p}$ (the probabilities of each outcome) for $\mathbf{x}$ (the number of times each outcome was observed) or vice versa. These two vectors have completely different natures: the first is an `acomp` composition from a real D-part simplex, and the second is a `ccomp` composition from a discrete (D, N)-simplex. These two vectors are as different as Figs. 3.3 and 3.4 are.

3.2.1.1 Simulation and Testing for Multinomial Distributions

Simulation of and testing for a multinomial distribution can be done with the following commands:

```
> (x <- rmultinom.ccomp(n=10,p=acomp(c(1,2,3)),N=8))
```

```
      [,1] [,2] [,3]
 [1,]    1    3    4
 [2,]    0    4    4
 [3,]    0    2    6
 [4,]    1    3    4
 [5,]    0    1    7
 [6,]    3    2    3
 [7,]    1    3    4
 [8,]    0    0    8
 [9,]    1    2    5
[10,]    0    3    5
attr(,"class")
[1] "ccomp"
```

```
> ccompMultinomialGOF.test(x)
```

```
    Pearson's Chi-squared test with simulated p-value (based
    on 1999 replicates)

data:  unclass(x)
X-squared = 21.23, df = NA, p-value = 0.2785
```

The multinomial goodness of fit test assumes the number of observations for each observation to be a possible varying constant; i.e., it tests conditional to the observed totals. The test is the χ^2-test for independence in a contingency table applied to the dataset.

3.2.2 The Multi-Poisson Distribution

The *multi-Poisson distribution* is the joint distribution of D-independent Poisson-distributed discrete random variables. Its parameter set is a vector of D rates $\boldsymbol{\lambda}$. If a vector $\mathbf{X}$ of D natural numbers has a multi-Poisson distribution, denoted as $\mathbf{X} \sim \mathscr{P}o^D(\boldsymbol{\lambda})$, then its density is

$$f(\mathbf{x};\boldsymbol{\lambda}) = \exp\left(-\sum_{i=1}^{D}\lambda_i\right)\prod_{i=1}^{D}\frac{\lambda_i^{x_i}}{x_i!}.$$

In this distribution, the components of $\mathbf{x}$ give the number of times that each of D possible outcomes A_i was observed in a given period, which expected rates of

occurrence during that period are the several λ_i. The expectation vector is the vector of expectations of the marginal Poisson distributions; i.e., $\mathrm{E}[\mathbf{X}] = \boldsymbol{\lambda}$, whereas the variance is a diagonal matrix with these very same elements, $\mathrm{var}[\mathbf{X}] = \mathrm{diag}(\boldsymbol{\lambda})$.

A look at the densities of the multinomial and the multi-Poisson distributions shows that they are pretty similar. There is only one difference between them, namely, the total sum of trials in a multinomial distribution is fixed, $\sum_{i=1}^{D} x_i = N$, whereas it is Poisson distributed in the multi-Poisson case. This similarity provides an interpretation of $\mathbf{p} = \mathscr{C}[\boldsymbol{\lambda}]$ as the vector of relative probabilities of occurrence of the D outcomes considered.

3.2.2.1 Simulation and Testing for a Multi-Poisson Distribution

Simulation of and testing for a multi-Poisson distribution can be done with the following commands:

```
> (x <- rpois.ccomp(n = 10, p = acomp(c(1, 2, 3)), lambda = 8))

      [,1] [,2] [,3]
 [1,]    3    3    8
 [2,]    1    3    4
 [3,]    2    3    3
 [4,]    1    1    0
 [5,]    0    2    7
 [6,]    1    1    4
 [7,]    3    2    3
 [8,]    0    3    4
 [9,]    2    0    1
[10,]    1    3    1
attr(,"class")
[1] "ccomp"

> ccompPoissonGOF.test(x)

    Count Composition Poisson Goodness of fit test

data:  structure(c(3, 1, 2, 1, 0, 1, 3, 0, 2, 1, 3, 3, 3, 1, 2,
   1, 2,  3, 0, 3, 8, 4, 3, 0, 7, 4, 3, 4, 1, 1),
   .Dim = c(10L, 3L), class = "ccomp")
= 0.5165, shape1 = 1, shape2 = 2, p-value = 0.7662
alternative hypothesis: Count Composition is not a multi-Poisson
 distribution with constant mean
```

The test for a multi-Poisson distribution reports the corrected p-value for the minimum of two p-values: one is the multinomiality test, which tests that conditional to the totals the individual count compositions follow a multinomial

distribution. The second is a Kolmogorov–Smirnov-based test for the Poisson distribution of the totals. Independence of the total and the conditional distribution is not tested.

3.2.3 Double Stochastic Count Distributions

The preceding two models for count compositions assumed a constant probability or intensity vector. That condition can be relaxed by assuming a hierarchical two-layer model. In the deeper layer, one may have a random Aitchison-distributed continuous composition $\mathbf{p} \sim \mathscr{A}(\boldsymbol{\theta}, \boldsymbol{\beta}_v)$, including as special cases the ALN and the Dirichlet distributions. In the second layer one has a random count composition $\mathbf{X}$, which conditional distribution is multinomial, for a known $\mathbf{p}$. This second layer is the one observed. The non-conditional density of $\mathbf{X}$, integrating all the uncertainty of both layers, is

$$f(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\beta}_v, N) = \frac{N! e^{-\kappa(\boldsymbol{\theta}, \boldsymbol{\beta}_v)}}{\prod_{i=1}^{D} x_i!} \times \int_{\mathbb{S}^D} \exp\left[(\boldsymbol{\theta} + \mathbf{x} - \mathbf{1}) \log(\mathbf{p})^t + \operatorname{ilr}(\mathbf{p}) \boldsymbol{\beta}_v \operatorname{ilr}(\mathbf{p})^t\right] \mathrm{d}\lambda_{\mathbb{S}}(\mathbf{p}),$$

at the nodes of the (D, n)-*simplex* and zero everywhere else (as the multinomial). Note that the term inside the integral is also an Aitchison density (3.3), except for the lack of the closure constant. Then, by adding and subtracting $\kappa(\boldsymbol{\theta} + \mathbf{x}, \boldsymbol{\beta}_v)$ inside that exponent and recalling that any density integrates to one, we find an alternative expression for the non-conditional density:

$$f(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\beta}_v, N) = \frac{N!}{\prod_{i=1}^{D} x_i!} \cdot \frac{e^{\kappa(\boldsymbol{\theta} + \mathbf{x}, \boldsymbol{\beta}_v)}}{e^{\kappa(\boldsymbol{\theta}, \boldsymbol{\beta}_v)}},$$

as a function of the closing constants of the Aitchison distribution. Thus, probabilities of the double stochastic count distribution can be quite easily computed by calling twice the function `AitchisonDistributionIntegrals`.

A double stochastic count distribution can be generated by doing a Poisson or multinomial simulation based on a random sample of compositions:

```
> vr = ilrvar2clr(diag(2))
> x1 <- rAitchison(10,theta=c(1,1,1),sigma=vr)
> rpois.ccomp(nrow(x1),x1,lambda=8)
```

```
     [,1] [,2] [,3]
[1,]    0    5    2
[2,]    1    1    5
[3,]    4    0    2
[4,]    0    2    1
[5,]    0    0    4
```

```
 [6,]    1    0   11
 [7,]    1    2    1
 [8,]    4    0    0
 [9,]    1    4    2
[10,]    2    0    1
attr(,"class")
[1] "ccomp"
```

```
> rmultinom.ccomp(nrow(x1),x1,N=8)
```

```
      [,1] [,2] [,3]
 [1,]    1    6    1
 [2,]    1    2    5
 [3,]    6    0    2
 [4,]    0    6    2
 [5,]    2    2    4
 [6,]    2    1    5
 [7,]    5    2    1
 [8,]    4    1    3
 [9,]    2    5    1
[10,]    6    0    2
attr(,"class")
[1] "ccomp"
```

There are no tests for goodness of fit for a double stochastic distribution in **R** or "`compositions`".

3.3 Relations Between Distributions

3.3.1 Marginalization Properties

3.3.1.1 Some Accessory Matrices

When the number of parts of a composition is higher than 3, we need a marginalization process to plot any of the preceding models. Three options can be considered: the construction of subcompositions, the amalgamation of some parts, and the geometric averaging of some parts. Proofs of results regarding amalgamation and subcomposition can be found in Aitchison (1986).

To characterize these properties, it is convenient to introduce two kinds of matrices: subcomposition matrices and amalgamation matrices. Both contain only zeroes and ones and explain in which way the parts of the original composition are kept, removed, or pooled together in the marginalization process. An *amalgamation matrix* **A** has D rows and some less columns, for instance A: in any row there must be *one single* 1, and every column must have *at least one* 1. A *subcomposition matrix* **B** has D rows and as many columns as parts in the subcomposition, say B: in any

row, there can be *at most one* 1, and every column must have *one single* 1. A third kind, the *geometric averaging matrix* **G**, is like an amalgamation matrix but where each column is divided by its total sum: in this way, the A columns of **G** sum up to one, and each row has one single nonzero value. Note that **A** and **B** denote two matrices, and A and B denote their number of columns, respectively.

Amalgamation and geometric averaging are, respectively, provided by the functions `rcompmargin` and `acompmargin`, where we specify the variables that should be kept unchanged, and the remaining variables are pooled together.

Example 3.3 (Amalgamation and subcomposition matrices). Take the example of nutrient composition of canned products introduced in Chap. 2, and replace fat by saturated and unsaturated fats (thus having 6 parts). Assume we want to either combine them back in a single variable or focus our attention only on them. Define the following three matrices:

$$\mathbf{A} = \begin{pmatrix} 1&0&0&0&0\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 1&0\\ 0&1\\ 0&0\\ 0&0\\ 0&0\\ 0&0 \end{pmatrix}, \quad \text{and} \quad \mathbf{G} = \begin{pmatrix} 1/2&0&0&0&0\\ 1/2&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix}.$$

If the way fat is split in its different types is not relevant for our problem and we want to replace them by the total fat, we can use the amalgamation matrix **A**. When applied to the data $\mathbf{X} \cdot \mathbf{A}$, this matrix replaces the first two columns of the dataset by their sum and keeps the other four unchanged (thus, the final number of columns $A = 5$). In the log-ratio perspective, amalgamation can be replaced by geometric averaging, which is obtained by applying **G** to the clr-transformed dataset: $\text{clr}^{-1}(\text{clr}(\mathbf{X}) \cdot \mathbf{G})$.

On the contrary, if the focus of our study is on fat components, we can use the subcomposition matrix **B**. Note that this matrix is obtained just by selecting some columns of an identity matrix, those linked to the parts that we want to keep. When applied to the data $\mathbf{X} \cdot \mathbf{B}$, this matrix selects $B = 2$ columns and removes the other from the dataset.

Another useful matrix is the matrix of projection onto the clr-plane of P components, $\mathbf{H}_P = \mathbf{I}_P - \frac{1}{P}\mathbf{1}_{P\times P}$. For instance, for $P = 4$

$$\mathbf{H}_4 = \begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 3&-1&-1&-1\\ -1&3&-1&-1\\ -1&-1&3&-1\\ -1&-1&-1&3 \end{pmatrix}.$$

Table 3.1 Marginalization properties of the distribution of a composition

$\mathbf{X}$ follows	$\mathbf{X} \cdot \mathbf{A}$ follows	$\mathbf{X} \cdot \mathbf{B}$ follows	$\text{clr}^{-1}(\text{clr}(\mathbf{X}) \cdot \mathbf{G})$ follows
$\mathcal{N}_{\mathbb{S}}(\mathbf{m}, \boldsymbol{\Sigma})$	(Unknown)	$\mathcal{N}_{\mathbb{S}}(\mathbf{m} \cdot \mathbf{B}, \mathbf{H}_B \cdot (\mathbf{B}^t \cdot \boldsymbol{\Sigma} \cdot \mathbf{B}) \cdot \mathbf{H}_B)$	$\mathcal{N}_{\mathbb{S}}(\text{clr}^{-1}(\text{clr}(\mathbf{m}) \cdot \mathbf{G}), \mathbf{H}_A \cdot (\mathbf{G}^t \cdot \boldsymbol{\Sigma} \cdot \mathbf{G}) \cdot \mathbf{H}_A)$
$\mathcal{D}i(\boldsymbol{\alpha})$	$\mathcal{D}i(\boldsymbol{\alpha} \cdot \mathbf{A})$	$\mathcal{D}i(\boldsymbol{\alpha} \cdot \mathbf{B})$	(Unknown)
$\mathcal{M}u(\mathbf{p}, N)$	$\mathcal{M}u(\mathbf{p} \cdot \mathbf{A}, N)$	(Mixture $\mathcal{B}i - \mathcal{M}u$)	(Unknown)
$\mathcal{P}o^D(\boldsymbol{\lambda})$	$\mathcal{P}o^A(\boldsymbol{\lambda} \cdot \mathbf{A})$	$\mathcal{P}o^B(\boldsymbol{\lambda} \cdot \mathbf{B})$	(Unknown)

3.3.1.2 Marginalization Properties of the Compositional Distributions

If $\mathbf{X}$ is a D-part random composition following a normal distribution on the simplex with geometric center $\mathbf{m}$ and clr variance $\boldsymbol{\Sigma}$, then a B-part subcomposition has also the same distribution model. The new geometric center is obtained by reclosing the selected parts from the original one, $\mathcal{C}[\mathbf{m} \cdot \mathbf{B}]$. The clr variance is obtained by taking the rows and columns related to the selected parts from the original variance matrix and forcing them to sum up to zero, i.e., with $\mathbf{H}_B \cdot (\mathbf{B}^t \cdot \boldsymbol{\Sigma} \cdot \mathbf{B}) \cdot \mathbf{H}_B$. If instead of removing the nonselected parts, we amalgamate them as a new part of the composition, and then the resulting composition has no known distribution. But if the nonselected parts are geometrically averaged, then we obtain again a normal distribution on the simplex, with parameters equivalent to those obtained with the subcomposition procedure: the mean is $\text{clr}^{-1}[\text{clr}(\mathbf{m}) \cdot \mathbf{G}]$ and the variance matrix $\mathbf{H}_A \cdot (\mathbf{G}^t \cdot \boldsymbol{\Sigma} \cdot \mathbf{G}) \cdot \mathbf{H}_A$.

If $\mathbf{X}$ has a Dirichlet distribution with parameter vector $\boldsymbol{\alpha}$, then a subcomposition has the same distribution with parameter vector $\boldsymbol{\alpha} \cdot \mathbf{B}$. The same occurs with the amalgamated marginal, which has a Dirichlet distribution of parameters $\boldsymbol{\alpha} \cdot \mathbf{A}$. Note that if the number of final parts is two, then the resulting composition has a Beta distribution. The marginal obtained by geometric averaging has no known distribution.

A count composition $\mathbf{X}$ with a multinomial distribution of parameters $\mathbf{p}$ and N does not have simple subcompositional properties: the distribution of a subset of parts (a non-reclosed subcomposition) can be obtained as a binomial mixture of multinomials, but this expression is beyond the scope of this book. On the contrary, an amalgamated count composition has also a multinomial distribution with parameters $\mathbf{p} \cdot \mathbf{A}$ and N. Note that if the number of final parts is two, then the resulting composition has a binomial distribution. The geometric averaging marginal does not have a known distribution.

For a multi-Poisson distribution of parameter $\boldsymbol{\lambda}$, both the amalgamation and extraction of (non-reclosed) subcompositions have nice properties, as the resulting distributions are of the same type, with parameter vector $\boldsymbol{\lambda} \cdot \mathbf{A}$ respectively $\boldsymbol{\lambda} \cdot \mathbf{B}$.

To our knowledge, the Aitchison distribution does not have any simple marginalization property. After looking at the summary of Table 3.1, that should not be a surprise: the properties of the Dirichlet and the normal distribution in the simplex are too different to allow for a common, general ground. The same occurs to the double stochastic count distribution, inheriting from the Aitchison distribution.

3.3.2 Conjugated Priors

This section summarizes some properties and relations of the preceding distributions that can be useful for Bayesian statisticians. As it is well known, the multinomial and the Dirichlet models are conjugate distributions. Assume a prior Dirichlet distribution with parameters $\boldsymbol{\alpha}$ for the vector $\mathbf{p}$. Update it by the likelihood of obtaining an outcome equal to the vector $\mathbf{x}$ from a multinomial distribution of probabilities $\mathbf{p}$. Then, the posterior distribution for $\mathbf{p}$ is also a Dirichlet distribution with parameters $(\boldsymbol{\alpha} + \mathbf{x})$.

The same happens between the multi-Poisson and the Dirichlet distributions. Take a prior model $\mathscr{C}[\boldsymbol{\lambda}] \sim \mathscr{P}o^D(\boldsymbol{\alpha})$ and a scaling value a (fixed or random). If we observed a multi-Poisson experiment $\mathscr{P}o^D(a\boldsymbol{\lambda})$ and obtained a realization $\mathbf{x}$, then the posterior $\mathscr{C}[\boldsymbol{\lambda}] \sim \mathscr{P}o^D(\boldsymbol{\alpha} + \mathbf{x})$.

The Aitchison distribution, being a generalization of the Dirichlet, has also the two preceding properties. If we take a prior model of Aitchison type, with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, for the parameter vector of the multinomial distribution, then the posterior obtained after observing a multinomial realization $\mathbf{x}$ has also an Aitchison-type distribution, with the same $\boldsymbol{\beta}$ and an increased $\boldsymbol{\theta} + \mathbf{x}$. The same posterior is obtained if $\mathbf{x}$ is a multi-Poisson realization. Finally, because the normal distribution on the simplex is just a particular case of the Aitchison distribution, we can also say that a prior $\mathscr{N}_{\mathbb{S}}(\mathbf{m}, \boldsymbol{\Sigma})$ updated by a multinomial or multi-Poisson observation $\mathbf{x}$ yields an Aitchison posterior $\mathscr{A}\left(\sum_{i=1}^{D} x_i, \mathbf{m} \oplus \mathrm{clr}^{-1}[\boldsymbol{\Sigma} \cdot \mathbf{x}], \boldsymbol{\Sigma}\right)$. Note that this relation provides an interpretation of α as a sort of total amount of information provided by the observations.

Finally, the double stochastic count distribution is actually the *predictive* distribution of a multinomial count composition with probability parameter following a priori an Aitchison distribution.

References

Aitchison, J. (1985). A general class of distributions on the simplex. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, *47*(1), 136–146.

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. London: Chapman & Hall (Reprinted in 2003 with additional material by The Blackburn Press), 416 pp.

Aitchison, J., Mateu-Figueras, G., & Ng, K. W. (2004). Characterisation of distributional forms for compositional data and associated distributional tests. *Mathematical Geology*, *35*(6), 667–680.

Mateu-Figueras, G. (2003). *Models de distribució sobre el símplex*. Ph. D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

Mateu-Figueras, G., & Pawlowsky-Glahn, V. (2005). The Dirichlet distribution with respect to the Aitchison measure on the simplex—a first approach. In G. Mateu-Figueras & C. Barceló-Vidal (Eds.), *Compositional data analysis workshop – CoDaWork'05, Proceedings*. Universitat de Girona, ISBN 84-8458-222-1, http://ima.udg.es/Activitats/CoDaWork05/.

Pawlowsky-Glahn, V. (2003). Statistical modelling on coordinates. In S. Thió-Henestrosa & J. A. Martín-Fernández (Eds.), *Compositional data analysis workshop – CoDaWork'03, Proceedings*. Universitat de Girona, ISBN 84-8458-111-X, http://ima.udg.es/Activitats/CoDaWork03/.

Rizzo, M. L., & Szekely, G. J. (2008). *energy: E-statistics (energy statistics)*. R package version 1.1-0.

Szekely, G. J., & Rizzo, M. L. (2005). A new test for multivariate normality. *Journal of Multivariate Analysis*, *93*(1), 58–80.

Chapter 4
Descriptive Analysis of Compositional Data

Abstract The descriptive analysis of multivariate data is, in classical applications, mostly a univariate and bivariate description of the marginals. This is an inappropriate approach for compositional data, because the parts of a composition are intrinsically linked to each other: the dataset is multivariate in nature, not by decision of the analyst. Following the principle of working in coordinates and the golden rule of working with log ratios, we can structure a meaningful descriptive analysis of a compositional dataset in the following steps. In the first step, descriptive statistics for central tendency, global spread, and codependence are computed. The second step is a preliminary look at a set of predefined "marginals", i.e., some ternary diagrams, to uncover relations between more than two parts. The third step is the exploration of the codependence structure through the biplot, a joint optimal representation of the variables and of the observations of a composition. Biplots are nevertheless explained in Chap. 6, because of their connection with principal component analysis. One of the aims of all the preceding phases should be to select a reasonable set of projections, or the selection of a basis: if this is achieved, the fourth step should be a classical univariate analysis of each coordinate.

4.1 Descriptive Statistics

The descriptive statistics commands will be exemplified with a subcomposition from a geochemical dataset:

```
> library(compositions)
> GeoChemSed=read.csv("geochemsed.csv",header=TRUE,skip=1)
> x = acomp(GeoChemSed, parts=c("Cr","Zn","Pb"))
```

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_4, © Springer-Verlag Berlin Heidelberg 2013

4.1.1 Compositional Mean

Aitchison (1986) pointed out that *meaningful* statistics of central tendency, of spread, and of codependence should have some invariance properties with respect to the most reasonable manipulations of the dataset:

- Translating (perturbing with a constant composition) a dataset should not change the spread or codependence structure in any sense, but just translate the center.
- Rescaling (e.g., changing units of) a dataset should simply rescale the center, whereas having a quadratic influence on spread and codependence (because variances have the units of the dataset but squared).

For compositions, translation and scaling are perturbation and powering (see Sect. 2.5). One obtains then the following descriptive statistics, meaningful with respect to these operations.

The *center* or *compositional mean* of a dataset $\mathbf{X}$ with N observations and D parts is the composition

$$\bar{\mathbf{x}} = \frac{1}{N} \odot \bigoplus_{n=1}^{N} \mathbf{x}_n = \mathrm{clr}^{-1}\left(\frac{1}{N}\sum_{n=1}^{N} \mathrm{clr}(\mathbf{x}_n)\right) = \mathscr{C}\left[\exp\left(\frac{1}{N}\sum_{n=1}^{N} \ln(\mathbf{x}_n)\right)\right], \tag{4.1}$$

where $\mathbf{x}_n$ is the nth observation, in the nth row of $\mathbf{X}$. Computing the `mean(x)` of an `acomp` object directly yields this compositional average (this uses standard S3-functionality of **R**).

```
> x = acomp(x)
> mean(x)
```

```
    Cr     Zn     Pb
0.1508 0.5773 0.2719
attr(,"class")
[1] acomp
```

In real multivariate analysis it is typical to center the data by removing their mean: in compositional data analysis, we can do the same by perturbing by the inverse of the center, i.e., $\mathbf{X}^* = \mathbf{X} \ominus \bar{\mathbf{x}}$:

```
> mean(x-mean(x))
```

```
    Cr     Zn     Pb
0.3333 0.3333 0.3333
attr(,"class")
[1] acomp
```

The average of the centered dataset is the neutral element of the simplex $\mathbb{1}$ which is a vector with the same value in each component.

Note that this estimator is not robust. A robust improvement of the compositional mean can be computed using `mean(x,robust=TRUE)`. Note also that, involving logs or products, this average does not admit a single zero value: this inability to manage zeros is one of the critical points usually raised against compositional statistics. How to work with zero values and outliers, as well as robustness issues, are discussed in Sect. 7.

4.1.2 Metric Variance and Standard Deviation

There are various measures of spread for compositional data. As a global measure of spread, one can use the *metric variance* (Pawlowsky-Glahn and Egozcue, 2001), also known as *total variance* or *generalized variance*,

$$\text{mvar}(\mathbf{X}) = \frac{1}{N-1} \sum_{n=1}^{N} d_A^2 (\mathbf{x}_n, \bar{\mathbf{x}}) , \tag{4.2}$$

i.e., the average squared distance from the center to the dataset, with corrected degrees of freedom as happens with the conventional variance. This variance is available through the generic command `mvar(x)`, as soon as `x` has `acomp` class:

```
> mvar(x)

[1] 0.2883
```

In real data, it is difficult to interpret variances quantitatively, and one tends to work with standard deviations. Unfortunately, there is no straightforward definition of a compositional standard deviation. We decided thus to define a *metric standard deviation* , to support quantitative interpretation, as

$$\text{msd}(X) = \sqrt{\frac{1}{D-1} \text{mvar}(X)}$$

```
> msd(x)

[1] 0.3797
```

If the variation was the same in all directions, we could interpret this quantity as the radial standard deviation on a log scale. When the variance is not the same in all directions, we may still understand the metric standard deviation as some sort of average spread.

As required by the principle of subcompositional coherence, the metric variance of a subcomposition is never greater than the metric variance of the parent composition.

4.1.3 Variation Matrix and Its Relatives

The metric variance does not contain any information about the codependence of components. According to the findings of Chayes (1975), we cannot describe these by raw correlations or covariances, because they are spurious effects of the closure. Instead Aitchison (1986) recommends the variation matrix. This matrix has D^2 components, each defined as

$$\tau_{ij} = \mathrm{var}\left(\ln \frac{x_i}{x_j}\right) \tag{4.3}$$

and estimated by

$$\hat{\tau}_{ij} = \frac{1}{N-1} \sum_{n=1}^{N} \ln^2 \frac{x_{ni}}{x_{nj}} - \ln^2 \frac{\bar{x}_i}{\bar{x}_j}$$

through the command

```
> variation(x)
```

```
       Cr     Zn     Pb
Cr 0.0000 0.2611 0.3259
Zn 0.2611 0.0000 0.2781
Pb 0.3259 0.2781 0.0000
```

It is easy to show that this is a symmetric matrix, given that $\ln(a/b) = -\ln(b/a)$ and $\mathrm{var}(-c) = \mathrm{var}(c)$. A small $\tau_{ij} = \tau_{ji}$ implies a small variance of the $\ln(x_i/x_j)$, thus a "good proportionality" $x_i \propto x_j$. The smaller a variation element is, the better the proportionality between the two components. To help in its interpretation, Aitchison (1997) suggests to consider the transformation $\rho_{ij} = \exp(-\tau_{ij}^2/2)$ and interpret it as a correlation coefficient. Figures 4.1 and 4.2 visualize several densities of an additive-logistic-normal distribution of two parts, with fixed center and different variations.

The same idea of applying a measure of location or spread to $\ln(x_i/x_j)$ can be used with any descriptive statistic. This is the result of the summary command for "acomp"-compositions:

```
> summary(x)
```

```
$mean
    Cr     Zn     Pb
0.1508 0.5773 0.2719
attr(,"class")
[1] acomp
```

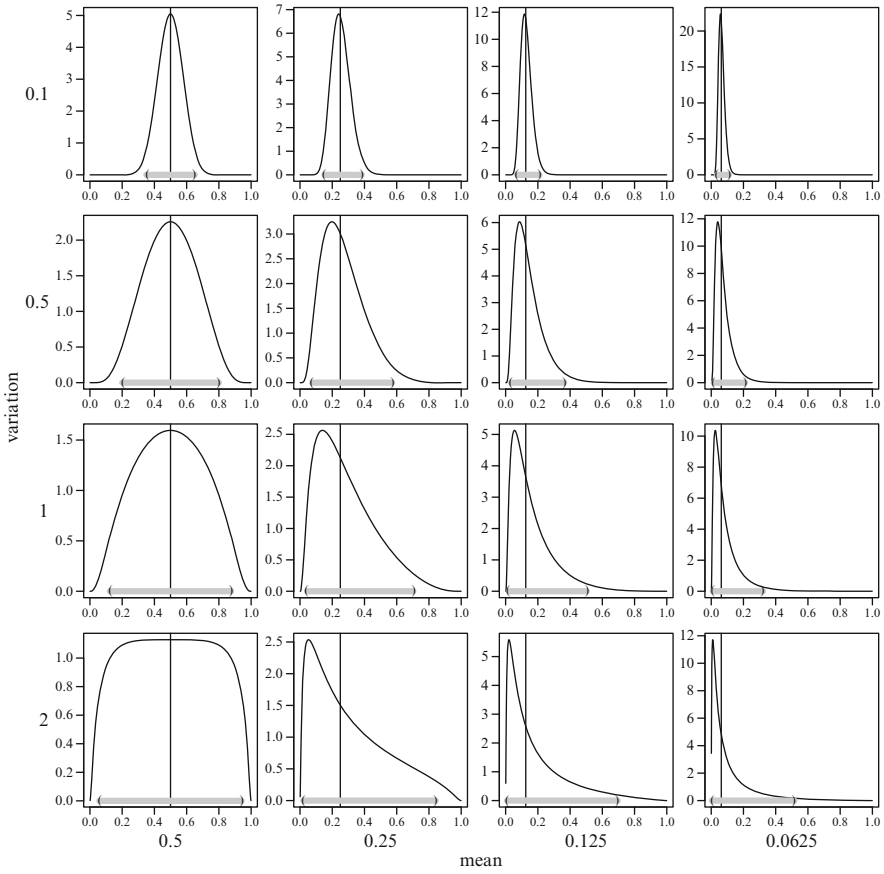


Fig. 4.1 Visualizing the meaning of the entries of the variation matrix. Each frame shows the additive-logistic-normal distribution density of a two-part composition with the given variation and mean. The *vertical line* marks the compositional mean, coinciding with the median. The mode is distorted because the density is not compatible with a compositional geometry (see Sect. 3.1.1). Symmetric 95 % probability intervals are shown in the X-axis

```
$mean.ratio
      Cr     Zn     Pb
Cr 1.000 0.2613 0.5548
Zn 3.827 1.0000 2.1232
Pb 1.803 0.4710 1.0000

$variation
       Cr     Zn     Pb
Cr 0.0000 0.2611 0.3259
Zn 0.2611 0.0000 0.2781
Pb 0.3259 0.2781 0.0000
```

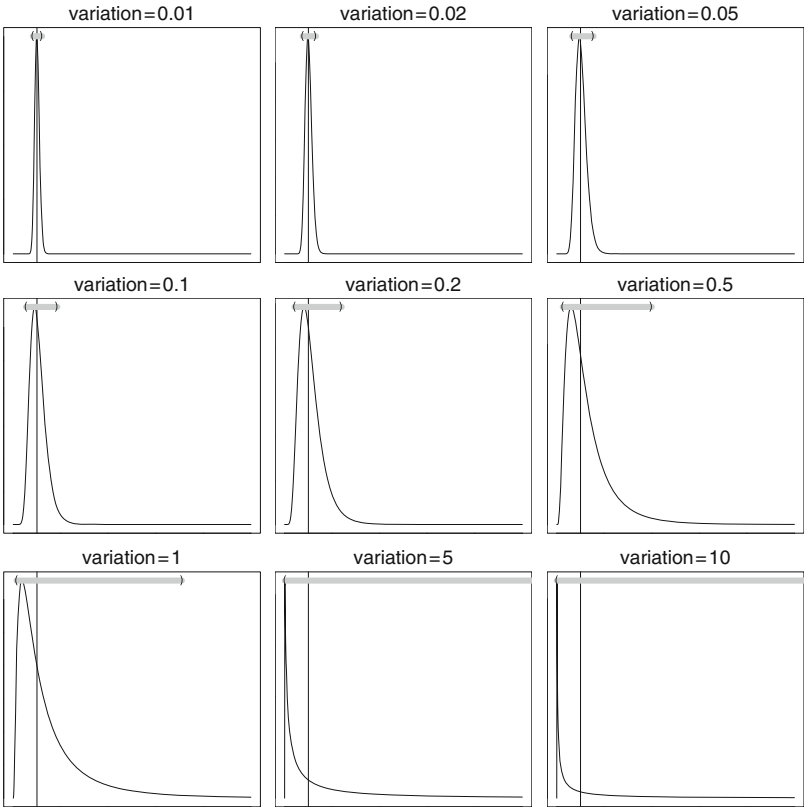


Fig. 4.2 Visualizing the meaning of the entries of the variation matrix. Each frame shows the density of the multiplicative factor `expsd`, by which the ratio of two components can be multiplied/divided to build a 1σ-interval. The pictures are based on a lognormal model. The *vertical line* marks the median and geometric mean. The mode is distorted because the lognormal density does not correctly capture the compositional geometry. Symmetric 95 % probability intervals are shown in the *upper part* of the plot

```
$expsd
      Cr    Zn    Pb
Cr 1.000 1.667 1.770
Zn 1.667 1.000 1.694
Pb 1.770 1.694 1.000

$min
      Cr      Zn      Pb
Cr 1.000 0.03571 0.05556
Zn 1.484 1.00000 1.06667
Pb 0.541 0.12500 1.00000
```

```
$q1
      Cr     Zn     Pb
Cr 1.000 0.1917 0.4286
Zn 2.619 1.0000 1.4545
Pb 1.310 0.3515 1.0000

$med
      Cr     Zn     Pb
Cr 1.000 0.2857 0.5455
Zn 3.500 1.0000 1.8125
Pb 1.833 0.5517 1.0000

$q3
      Cr     Zn     Pb
Cr 1.000 0.3819 0.7639
Zn 5.215 1.0000 2.8454
Pb 2.333 0.6875 1.0000

$max
   Cr     Zn    Pb
Cr  1 0.6739 1.848
Zn 28 1.0000 8.000
Pb 18 0.9375 1.000

$missingness
        missingType
variable NMV BDT MAR MNAR SZ Err
      Cr  87   0   0    0  0   0
      Zn  87   0   0    0  0   0
      Pb  87   0   0    0  0   0

attr(,"class")
[1] "summary.acomp"
```

- `mean.ratio` is the geometric mean of the ratios and can be interpreted as a central value of each pairwise ratio.
- `expsd` is the exponential of the standard deviation of the log ratios, undercase, to be taken as a factor to multiply/divide the mean.ratio of the components and obtain a 1σ-radius interval (Fig. 4.2 presents a visual evaluation of how large or small these values might be).
- Finally, for each pairwise ratio, we obtain the minimum (`min`), the first quartile (`q1`), the median (`median`), the third quartile (`q3`), and the maximum `max`.

With all these quantities we can get a good feeling of the relative variability of the components. But an image is worth a thousand words, and all these statistics are graphically displayed by the boxplot. The command `boxplot(x)` of an "`acomp`"

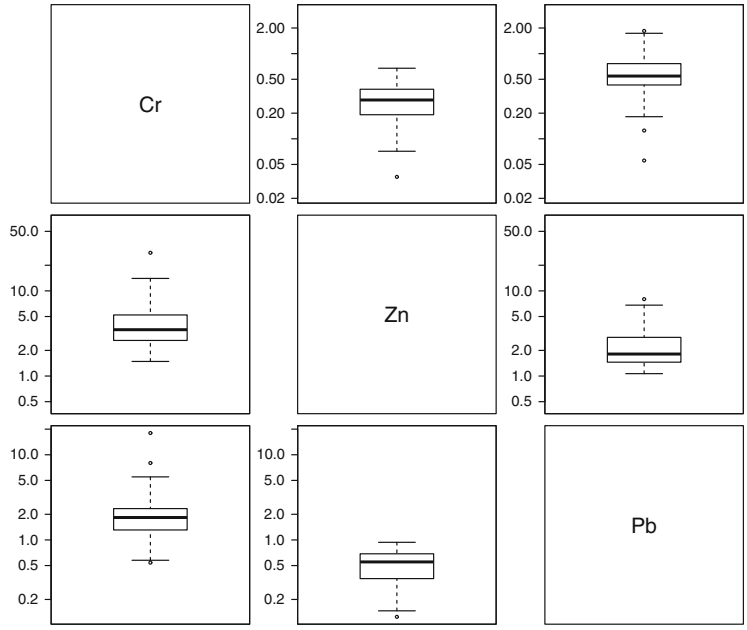


Fig. 4.3 Boxplots of all pairwise log ratios. This is the result of `boxplot(x)` with an `acomp` object

compositional data generates a matrix of boxplots, where each cell displays the boxplot of a pairwise log ratio (Fig. 4.3):

```
> opar <- par(mar=c(3,3,1,1))
> boxplot(x)
> par(opar)
```

4.1.4 Variance Matrices

Once the analyst is familiar with them, variation matrices are very practical for interpretation, because they involve only pairs of variables. On the contrary, some further computations will need a mathematical description of the codependence in a more classical way, as a a matrix of covariances. For these uses, we have the the *clr-variance matrix* (by the function `var`): it is just the variance-covariance matrix of the clr-transformed dataset. Its elements $\hat{\Sigma}_{ij}$ are thus the covariances

$$\hat{\Sigma}_{ij} = \mathrm{cov}\left(\mathrm{clr}_i(\mathbf{x}), \mathrm{clr}_j(\mathbf{x})\right) = \frac{1}{N}\sum_{n=1}^{N} \ln\frac{x_{ni}}{g(\mathbf{x}_n)} \cdot \ln\frac{x_{ni}}{g(\mathbf{x}_n)} - \ln\frac{\bar{x}_i}{g(\bar{\mathbf{x}})} \cdot \ln\frac{\bar{x}_j}{g(\bar{\mathbf{x}})}, \tag{4.4}$$

where $\bar{\mathbf{x}} = [\bar{x}_1, \ldots, \bar{x}_D]$ is the center (4.1). Remember that the components of the clr sum up to zero and so does the clr-variance matrix: each of its rows and of its columns sums up to zero. Moreover, each of these covariances involve *all* parts in its computation (through the factor $g(\mathbf{x})$). Furthermore, the clr-covariance matrix has again forced negative covariances. It should thus not be interpreted directly and *never be converted to a correlation matrix*. It is just a mathematical tool.

Some compositional procedures also use-at least internally-the *ilr-variance matrix*, computed with `clrvar2ilr(var(x))`. The ilr-variance matrix is not unique, since it depends on the selection of the basis. It should thus not be reported without explicitly mentioning which basis was used. Working with bases will be treated later in this chapter. The mathematical meaning of these variance matrices is that of a bilinear form, as was mentioned in Sect. 2.5.9.

Example 4.1 (Standardization, centering, and scaling). In many multivariate applications, it is typical to center and/or scale the several variables, in order to have a dimensionless, comparable variability. This is typically done by the *standardization*, i.e., *centering* by subtraction of the mean value and *scaling* by division with the standard deviation. These operations are flawed in compositional data, for several reasons. First, compositional data already share a common dimensionless scale; thus, scaling removes an important information on which variables present more variability. Second, centering produces negative values, which destroys the interpretation of results as compositions.

However, we can apply the equivalent operations of centering and scaling (thus a standardization) to a compositional dataset, by using perturbation and powering. A dataset can be centered by perturbation with the inverse of the center (4.1) and scaled by powering with the inverse of the square root of the metric variance (4.2):

$$\mathbf{Z} = \frac{1}{\sqrt{\text{mvar}(\mathbf{X})}} \odot (\mathbf{X} \ominus \bar{\mathbf{x}}) \,.$$

We can scale a compositional dataset, either by using this expression above or the function `scale`:

```
> # get the statistics:
> mn = mean(x)
> mvr = mvar(x)
> # center the data:
> dat1 = scale(x,center=TRUE,scale=TRUE)
> dat2 = (x-mn)/sqrt(mvr)
```

If we only want to center the data, we can try either

```
> dat3 = scale(x,center=TRUE,scale=FALSE)
> dat4 = x-mn
```

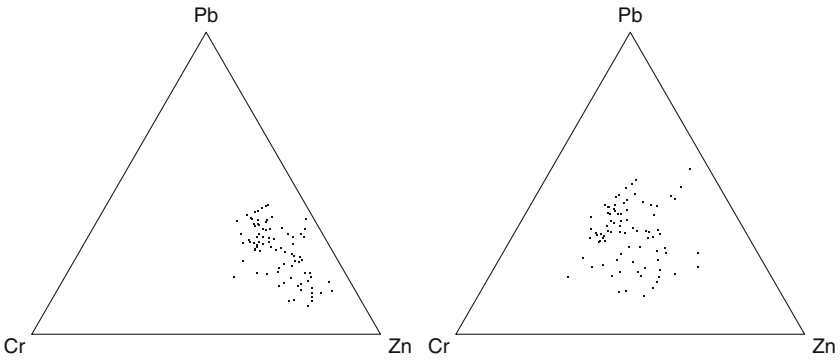


Fig. 4.4 Comparison of a ternary diagram of centered and non-centered data

It is left to the reader to check that these pairs of data are equal. Compositional scaling does not standardize each variable independently, but "powers" all of them by the same quantity. This preserves the different variances of each clr variable and just makes their total sum constant. A transformation like this is mostly useful to compare a subcomposition among several subsamples. On the contrary, centering is quite often used in plotting data: if we have data "squashed" on an edge or a vertex of the ternary diagram, we can enhance the visual assessment of its structure by centering them before plotting. For this reason, `plot.acomp()` can also be given logical arguments `center` and `scale`. Figure 4.4 compares the non-centered and the centered versions of this diagram.

4.1.5 Normal-Based Predictive and Confidence Regions

It is interesting to represent center and dispersion structure of the data graphically. The center, being a point, can be plotted by the function `plot(x,add=TRUE)`, where `x` must be the `acomp` mean. The dispersion structure can be visualized with `ellipses(mean,var,r)`. This ellipse can actually correspond to a p-confidence region on the mean, if we take the radius `r` according to a Fisher $\mathscr{F}$ distribution of $(D-1)$ and $(N-D+1)$ degrees of freedom:

$$r = \sqrt{\frac{D-1}{N-D+1} \cdot \mathscr{F}_p(D-1, N-D+1)}.$$

This is valid if the data follow a normal distribution for compositions (see Sect. 3.1), or else if the number of observations N is *large*. In this last case, a compositional central limit theorem applies (Aitchison, 1986).

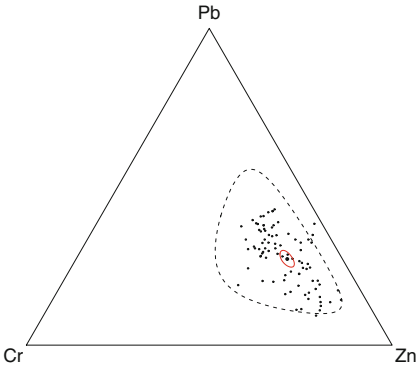


Fig. 4.5 Elliptic 95 % confidence region around the center (*solid line*) and 95 % probability region of the population (*dashed line*) in a ternary diagram (Example 4.2). The center is also represented as a *dot*

Alternatively, we can build p-probability regions, if we assume that the data follow a normal distribution for compositions *of known variance matrix*. Then, the ellipses must have a radius given by the χ^2 distribution of $(D-1)$ degrees of freedom:

$$r = \sqrt{\chi^2_\alpha (D-1)}.$$

Example 4.2 (Confidence and probability regions). Figure 4.5 shows both confidence and probability regions, for the dataset of Cr, Zn, and Pb of the geochemical sediment composition. These regions are obtained with the following steps. First, we compute the several statistics needed, including the radii of the two regions:

```
> # get the mean:
> mn = mean(x)
> # compute variance and radii:
> vr = var(x)
> df1 = ncol(x)-1
> df2 = nrow(x)-ncol(x)+1
> rconf = sqrt( qf(p=0.95, df1, df2)*df1/df2 )
> rprob = sqrt( qchisq(p=0.95, df=df1) )
```

and then plot the data, the center, and the two ellipses:

```
> # plot the data:
> opar = par(mar=c(3,3,1,1))
> plot(x, cex=0.25)
> plot(mn, pch=19, cex=0.5, add=TRUE)
> # plot the ellipses:
> ellipses(mean=mn, var=vr, r=rconf, col="red")
> ellipses(mean=mn, var=vr, r=rprob, lty=2)
> par(opar)
```

Example 4.3 (A complete descriptive analysis). The dataset `Hydochem` contains several dissolved ions analyzed in some highly polluted rivers. For now, we focus our attention on the main cations ($Na^+ + K^+$, Ca^{2+}, Mg^{2+}) and anions (Cl^-, HCO_3^-, $SO_4^=$):

```
> library("compositions")
> data("Hydrochem")
> subcomp = acomp( Hydrochem[,c(7:10,14,17,18)] )
> # average in mass proportions
> mean(subcomp)
```

```
     Na       K      Mg      Ca      Cl     SO4    HCO3
0.10686 0.01203 0.03800 0.14119 0.15875 0.22932 0.31385
attr(,"class")
[1] acomp
```

This mean shows us that the ion mass proportion is in average dominated by the cations Na and Ca, whereas anions show a fair equilibrated contribution. But in fact, molar proportions would be more informative; thus, we recast the data by inverse-perturbing them with the molar weights of each ion:

```
> # weight of a mol of each species:
> mw = c(22.99, 39.098, 34.305, 40.078, 35.453, 96.063, 61.017)
> mw = acomp(mw)
> # recast mass proportions to molar proportions
> subcomp = subcomp-mw
> mean(subcomp)
```

```
     Na       K      Mg      Ca      Cl     SO4    HCO3
0.21525 0.01424 0.05130 0.16314 0.20735 0.11054 0.23818
attr(,"class")
[1] acomp
```

The numbers have not changed a lot, though now Na dominates the ions, and we see a certain similarity between Na and Cl. A correct chemical analysis should require that there are as much negatively charged ions (anions) as positively charged ones (cations). By reweighting this average with the charge of each species, we obtain

```
> # ionic charges of each species:
> charges = c(1,1,2,2,-1,-2,-1)
> unclass(mean(subcomp)) %*% charges
```

```
          [,1]
[1,] -0.008248
```

which is a fairly low error, given that we are not considering the contribution of many more ions. Note that the instruction `unclass()` strips the object of its

class(`acomp` in this case) and allows us to treat it as a vector of real numbers, not as a composition.

A look at the variation matrix,

```
> variation(subcomp)
```

```
          Na      K     Mg      Ca      Cl    SO4    HCO3
Na   0.00000 0.4539 0.7443 1.11177 0.08632 0.8396 1.25979
K    0.45387 0.0000 1.1885 1.69726 0.41132 1.6289 1.69015
Mg   0.74429 1.1885 0.0000 0.21956 0.95977 0.2191 0.28328
Ca   1.11177 1.6973 0.2196 0.00000 1.36408 0.2001 0.08803
Cl   0.08632 0.4113 0.9598 1.36408 0.00000 1.0970 1.52865
SO4  0.83958 1.6289 0.2191 0.20008 1.09697 0.0000 0.39657
HCO3 1.25979 1.6901 0.2833 0.08803 1.52865 0.3966 0.00000
```

highlights again the very good codependence between Cl and Na, a very low 0.0863, as well as between HCO_3 and Ca, 0.088 (compare these values with Figs. 4.1 and 4.2 as a reference). This is readily explained by the existence of a very important common source for each one of these pairs, respectively leaching from some potash mines in the basin and the calcareous-dominated landscape. For the same reason, one should expect also a good codependence between Mg and both Ca and HCO_3 (provided by dolomite) and between Ca and SO_4 (constituents of gypsum), respectively, 0.2196, 0.2833, and 0.2001. It is surprising to see that K has no codependence with Na or Cl (as they are the main components of potash) but also not a good one with SO_4 (both being typical constituents of fertilizers).

4.2 Exploring Marginals

4.2.1 The Three Types of Compositional Marginals

Our inability to see in more than 3 dimensions (and to print in more than 2) has led to the standard practice of preliminarily looking at the set of scatterplots of each pair of available variables: thus, the standard outcome that one gets when doing `plot(x)` (for `x` a `data.frame`) is a matrix of scatterplots, where one represents the row variable in the Y-axis against the column variable in the X-axis (see Fig. 4.6). These bivariate scatterplots are flawed for compositions, as mentioned in Sect. 2.3.1.

For compositional data, the typical 2-dimensional representations are ternary diagrams, and this will be the result of function `plot(x)` for `acomp` objects with 3 parts. For compositions with more than 3 parts, the result is a matrix of ternary diagrams, involving the two variables defined by row and column. But to plot a

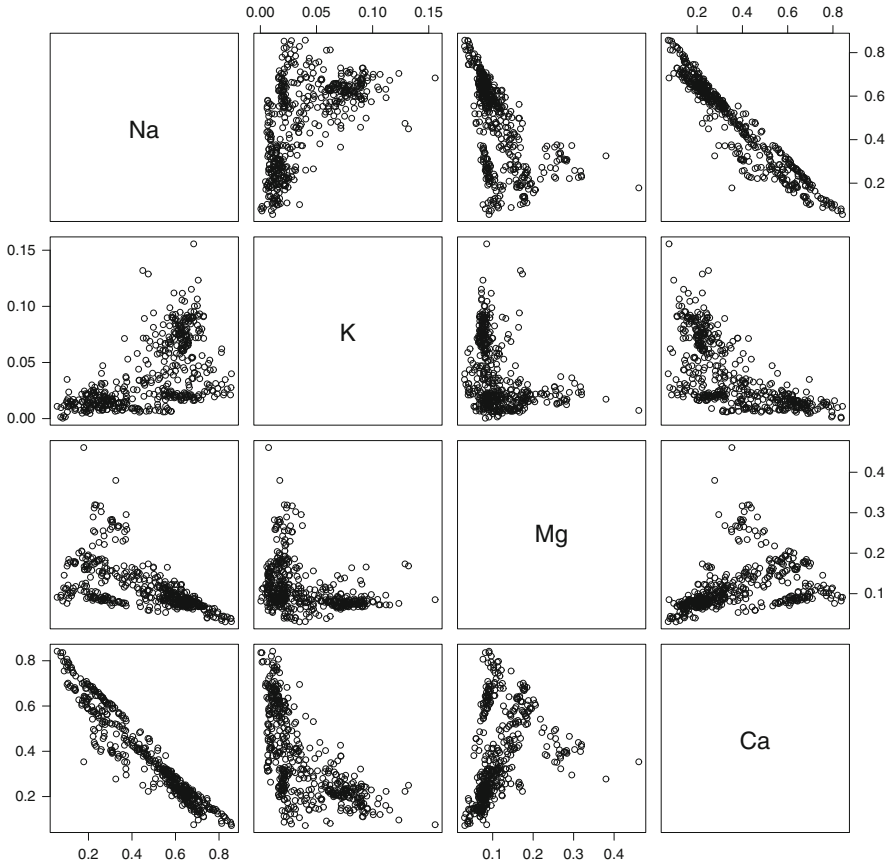


Fig. 4.6 Result of plotting a conventional data frame. Note that this is a matrix of the so-called Harker diagrams, thus with the problems mentioned in Sect. 2.3.1

ternary diagram, we need three variables. Thus, we will need to take a decision regarding the third component. Function `plot(x, margin)` generates a matrix of ternary diagrams, which three parts are the two defined by the row and the column and the third is controlled by the optional argument `margin`:

- `margin="acomp"` (or nothing, the default) computes the third part as the geometric mean of all the components except those two from row and column (symbolized with `"*"`; see Fig. 4.7).
- `margin="rcomp"` computes the third part as the amalgamation of all the components except those two from row and column (symbolized with `"+"`; see Fig. 4.8).
- Finally, `margin` equal to one of the parts of the dataset fixes it as the third part of the whole matrix and generates thus a matrix with $(D - 1)$ rows and columns (see Fig. 4.9).

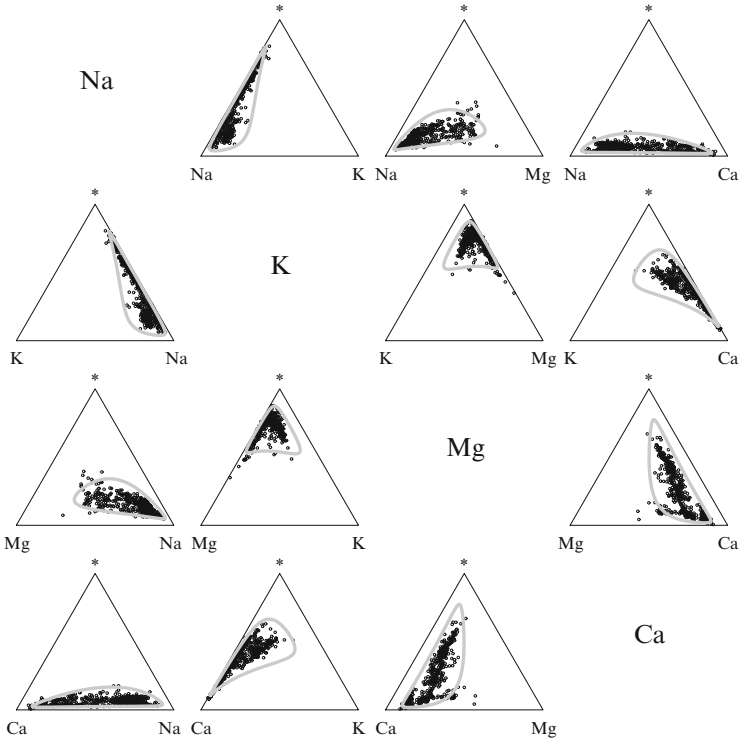


Fig. 4.7 Matrix plot of ternary diagrams with default third component (geometric average) and ellipses showing the variance structure through 95 % probability regions

The default behavior is `margin="acomp"` because it is consistent with the whole Aitchison geometry of the simplex explained in Sect. 2.5, allowing then to add further information (center, variance ellipses, principal component lines, etc.). The option `margin="rcomp"` is provided because amalgamation is a quite common technique of reduction of dimensionality used by many analysts. One should nevertheless be extremely careful when using amalgamation, as it is a nonlinear operation in the Aitchison compositional geometry and does not honor the principles of compositional analysis.

4.2.2 Ternary Diagram Matrices for Multidimensional Compositions

One of the advantages of using `margin="acomp"` is that one can apply many coherent manipulations to the dataset, like centering the dataset or graphically representing its variance-covariance structure (numerically obtained in Sect. 4.1), by adding ellipses to the plot:

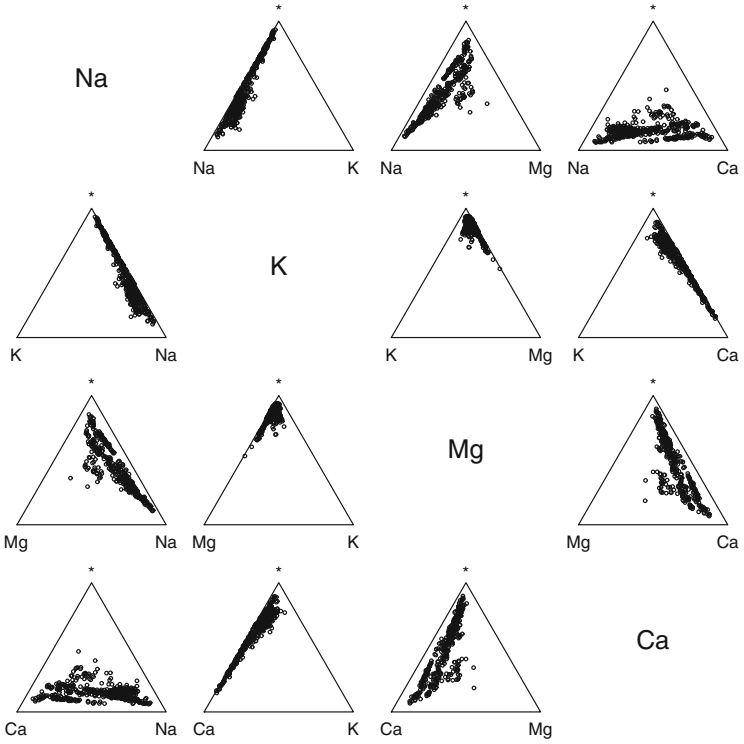


Fig. 4.8 Matrix plot of ternary diagrams with amalgamated third component

```
> cations = acomp( subcomp[,1:4] )
> plot(cations, cex=0.5)
> r = sqrt( qchisq(p=0.95,df=2) )
> mn = mean(cations)
> vr = var(cations)
> ellipses(mean=mn, var=vr, r=r, lwd=3 ,col="gray")#red?
```

The radius has been chosen to contain 95 % of probability assuming a normal model for the composition (see Sect. 3.1) and a known variance. Figure 4.7 shows the result. In contrast, if argument `margin="rcomp"` is given, one obtains Fig. 4.8. The third alternative is generated following this code, resulting in a set of ternary diagrams with third component fixed as Mg (Fig. 4.9, in this case, also centered giving argument `center=TRUE`):

```
> plot(cations, cex=0.5, center=TRUE, margin="Mg")
> mn = mean(cations)
> ellipses(mean=mn, var=vr, r=r, lwd=3,col="gray")#red?
```

From these diagrams, those of Ca vs. Mg vs. K and/or Na are the most interesting. First, they show a very narrow variance ellipse: the data seem to follow

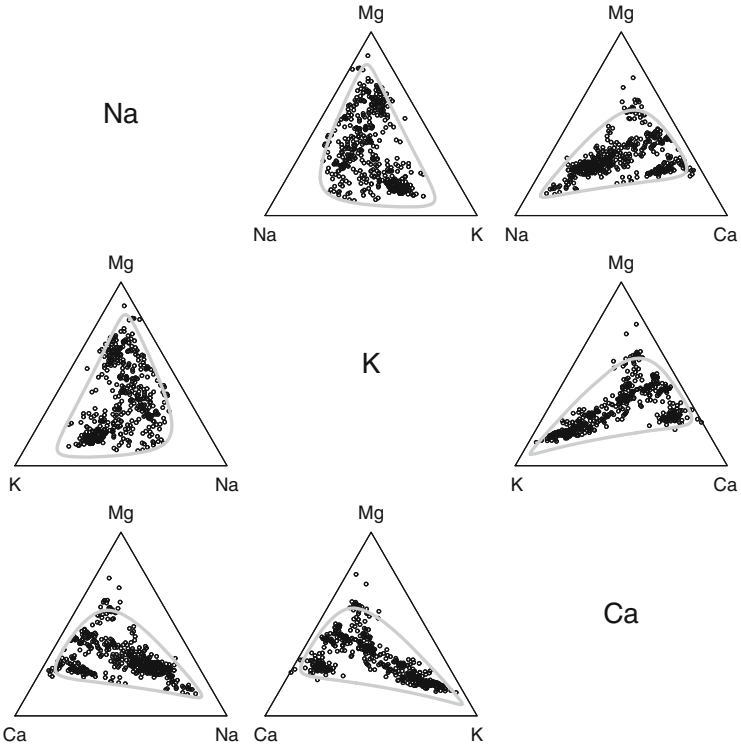


Fig. 4.9 Matrix plot of *centered* ternary diagrams with Mg fixed as third component and ellipses showing the variance structure through 95 % probability regions

a one-dimensional pattern joining the vertices Ca and K–Na. This is in fact a compositional linear pattern and can be estimated by a principal component technique (see Sect. 6.1.2). In this particular case, such a narrow ellipse tells us that the ratios involing Mg have almost a constant value, and most compositional variance is due to the variatiability of Na/Ca and K/Ca. Second, one can also see a differentiated subgroup, like a branch of data pointing towards Mg (see Sect. 6.2 for more information on group detection).

4.3 Exploring Projections

4.3.1 Projections and Balances

The *projection* of a composition onto a given compositional direction is a real, unbounded value (page 39). Therefore, such projections can be treated with any of the conventional methods of descriptive analysis: descriptive statistics, histograms,

boxplots, kernel density estimates, etc. The present subsection explains how to encode and compute certain projections in "`compositions`", in order to easily feed them further to these descriptive methods. But these real projections are also coordinates with respect to a basis. Thus, the following subsection introduces certain kinds of bases of easier interpretation and some ways to compute them.

The grounding concept here is the balance (Egozcue and Pawlowsky-Glahn, 2005). A *balance between two groups* of parts A and B is a measure of the total relative importance of one group against the other (as its name suggests). To be consistent with the whole compositional geometry of Sect. 2.5, that total relative importance must be a product, and the measure of contrast must be a log ratio:

$$\xi_{A/B} = K \ln \frac{\left(\prod_{a\in A} x_a\right)^{N_b}}{\left(\prod_{b\in B} x_b\right)^{N_a}} = K \sum_{a\in A} N_B \ln x_a - K \sum_{b\in B} N_A \ln x_b = \left(\mathbf{x}, \mathrm{clr}^{-1}\left(\boldsymbol{\beta}_{A/B}\right)\right) \tag{4.5}$$

with $K^{-1} = \sqrt{N_a N_b (N_a + N_b)}$. Note that with the trick of powering each group of parts by the number of elements of the *other* group (N_a and N_b), we ensure that the log-ratio coefficients sum up to zero. The constant K is a normalization to allow mutual comparison between balances with different number of parts. But a balance is also the projection of a composition onto a certain direction, the *balancing element*, $\mathrm{clr}^{-1}\left(\boldsymbol{\beta}_{A/B}\right)$, where the components of $\boldsymbol{\beta}_{A/B}$ are as follows: $\beta_i = +N_b/K$ if $i \in A$, $\beta_i = -N_a/K$ if $i \in B$, and $\beta_i = 0$ for those parts which are neither in A nor in B.

Balance Computation. Balance scores are computed with the command `balance(X, expr)`, where `X` is the dataset and `expr` a *formula* with the association of parts. For instance, the scores of the balance suggested in (6.2) would be obtained with the command

```
> balance(X=x, expr=~CaO/(Na2O*P2O5))
```

Balance elements (clr-transformed) are provided by the command `balanceBase(X, expr)`, with the same arguments. For instance,

```
> x = acomp(GeoChemSed,4:13)
> balanceBase(X=x, expr=~CaO/(Na2O*P2O5))

       CaO/Na2OP2O5
SiO2         0.0000
TiO2         0.0000
Al2O3        0.0000
MnO          0.0000
MgO          0.0000
CaO          0.8165
Na2O        -0.4082
```

```
K2O          0.0000
P2O5        -0.4082
Fe2O3t       0.0000
```

Note that in the formula expression, we *do not* include the numeric coefficients, but only the variables involved and their position (in the numerator, group A, or in the denominator, group B).

4.3.2 Balance Bases

Why is selecting a basis so important? The principle of working in coordinates more or less implies that we get the same result with any orthogonal basis. Being able to define a proper orthonormal basis is something instrumental, to be able to pass from the composition to its ilr coordinates and back. However, with a well-chosen basis, it might be possible to interpret individual coefficients and parameters on the level of coordinates. The black-box, default ilr coordinates (2.9)–(2.11) may be suitable for a quick-and-dirty analysis, but the interpretation of results in terms of the coordinates is mostly impossible. This is greatly eased if one uses a basis which has already a connection with the physical reality being described. This is not always possible, but when it is, it pays the effort of building ad hoc bases.

Note that by construction (4.5) a balance element has always a unit Aitchison norm. If we are able to define $(D-1)$ of them being additionally mutually orthogonal, then we found an orthonormal basis (page 39). Such balance bases are straightforward to define with the help of a *sequential binary partition*. First, we consider all available parts and split them in two groups. Then, we take one of these groups and split it again in two. This is applied recursively until all groups have one single part. The groups do not necessarily have similar number of parts: to actually get an interpretable set of ilr coordinates, the splitting should be based on previously known criteria of association or affinity. If such criteria are not clear, one can use a hierarchical cluster analysis (Sect. 6.2) and define a balance basis with the output (e.g., Fig. 6.7).

Encoding Balance Bases. There are several ways to describe a basis using balances: a partition formula, a signed-binary matrix, and a merging tree structure. *Partition formulae* are like the balance formulae of the preceding box, but involving all parts in a nested structure. *Sign-binary (signary) matrices* have the same shape than ilr basis matrices (D rows and $D-1$ columns), but only with values of $+1$, -1, and 0: the element (i,j) is $+1$ if the ith part is in the first group of jth step partition, or -1 if it is in the

second group, or it is 0 if it is not involved in that step. Finally, *merging tree structures* is the way **R** uses to encode dendrograms and cluster analysis results: these are matrices of $(D-1)$ rows and 2 columns, where each row represents an agglomeration step, and the two elements give the two agglomerated individuals or previously formed groups (see Sect. 6.2 and `? hclust` for details).

These elements will be illustrated with the example of Fig. 6.7. The partition formula is

```
( (Al2O3/K2O)/(SiO2/Na2O) ) /
   ( (CaO/P2O5) / (MgO/(TiO2/(MnO/Fe2O3t))) )
```

This can be passed in `expr` to `balanceBase(X, expr)`, resulting in a particular basis **V**. An equivalent basis will be obtained by processing the signary matrix given in this table:

Part		Step number								
Number	Name	1	2	3	4	5	6	7	8	9
1	SiO_2	0	0	1	0	0	0	−1	0	1
2	TiO_2	0	0	0	1	0	−1	0	−1	−1
3	Al_2O_3	0	1	0	0	0	0	1	0	1
4	MnO	1	0	0	−1	0	−1	0	−1	−1
5	MgO	0	0	0	0	0	1	0	−1	−1
6	CaO	0	0	0	0	1	0	0	1	−1
7	Na_2O	0	0	−1	0	0	0	−1	0	1
8	K_2O	0	−1	0	0	0	0	1	0	1
9	P_2O_5	0	0	0	0	−1	0	0	1	−1
10	$Fe_2O_3^t$	−1	0	0	−1	0	−1	0	−1	−1

The actual basis matrix **V** is obtained with the function `gsi.buildilrBase(W)`, where `W` is a signary matrix.

Once obtained the ilr basis matrix, this can be passed in argument `V` to the ilr-coordinate computation function `ilr(x, V)`. Alternatively, for the formula case we can also use the function `balance(X, expr)`, without explicitly computing the basis. Take into account that, though these three ways encode the "same" basis in practical terms, the ordering of the columns might be different between them. This may change the position of the coordinates in the ilr-transformed vector!

4.3.3 The Coda-Dendrogram

The last issue of this section is the representation of the chosen basis with the coda-dendrogram (Egozcue and Pawlowsky-Glahn, 2005). A *coda-dendrogram*

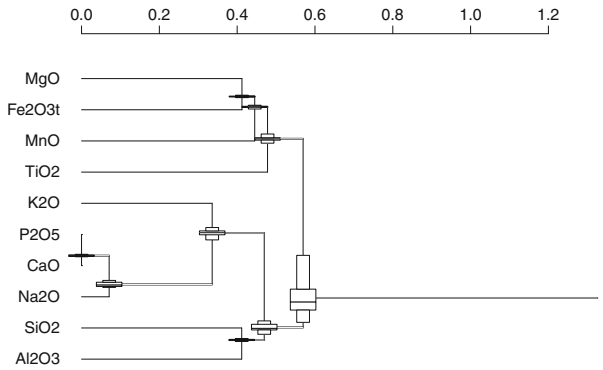


Fig. 4.10 Coda-dendrogram of the ilr basis for the geochemical composition of glacial sediments, generated by the clustering algorithm (Example 6.5). Each axis contains the interval (−4, 4), the default, corresponding to proportions of 0.35 and 99.65 % respectively

graphically shows in a hierarchy how the several parts are grouped together; thus, it displays the partition formula. Further, the segment joining two groups is taken as an axis for the corresponding balance. That axis is then used to mark the position of the balance mean, and a bar is drawn, with a length showing the balance variance. Even more, the axis may then be used to place any desired one-dimensional graphical summary of the dataset: the default is to represent box-plots, but one can also put kernel density estimates, histograms, or a rug of data. This representation (Fig. 4.10) is obtained with the command `CoDaDendrogram`, needing a compositional dataset (`X`) and a descriptor of the ilr basis chosen, which is one of the following: the basis matrix `V` itself, the `mergetree` structure, the `signary` matrix, or the partition formula through argument `expr`.

References

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. London: Chapman & Hall (Reprinted in 2003 with additional material by The Blackburn Press), 416 pp.

Aitchison, J. (1997). The one-hour course in compositional data analysis or compositional data analysis is simple. In V. Pawlowsky-Glahn (Ed.), *Proceedings of IAMG'97 — The third annual conference of the International Association for Mathematical Geology*, Volume I, II and addendum (pp. 3–35). Barcelona: International Center for Numerical Methods in Engineering (CIMNE), 1100 pp.

Chayes, F. (1975). A priori and experimental approximation of simple ratio correlations. In R. B. McCammon (Ed.), *Concepts in geostatistics* (pp. 106–137). New York: Springer, 168 pp.

Egozcue, J. J., & Pawlowsky-Glahn, V. (2005). Groups of parts and their balances in compositional data analysis. *Mathematical Geology*, *37*(7), 795–828.

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

Chapter 5
Linear Models for Compositions

Abstract Compositions can play the role of dependent and independent variables in linear models. In both cases, the parameters of the linear models are again compositions of the same simplex as the data. Most methods for classical linear models have a close analog in these compositional linear models. This chapter addresses several questions on this subject. What are compositional linear models? How to visualize the dependence of compositions, already multivariable, with further external covariables? How to model and check such dependence with compositional linear models? What are the underlying assumptions? How can we check these assumptions? What is the compositional interpretation of the results? How to use linear models to provide statistical evidence with tests, confidence intervals, and predictive regions? How to visualize model results and model parameters? How to compare compositional linear models and how to find the most appropriate one?

5.1 Introduction

To explain compositional regression, it is very helpful to have a common understanding of classical regression, its **R** interface, and how it is used.

5.1.1 Classical Linear Regression (Continuous Covariables)

5.1.1.1 The Univariate Case

A (non-compositional) linear regression models the dependence of one observed variable Y on another variable X by a simple equation:

$$Y_i = a + bX_i + \varepsilon_i,\ \varepsilon_i \sim N(0, \sigma^2).$$

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_5, © Springer-Verlag Berlin Heidelberg 2013

Y is called the dependent variable and X the independent variable. ε is an error term independent of X and independent between the different cases indexed by $i = 1, \ldots, n$. These errors are typically modeled with a normal distribution $\varepsilon_i \sim N(0, \sigma^2)$ with expectation 0 and an unknown but constant variance σ^2. We estimate the parameters by

$$\hat{b} = \frac{\widehat{\text{cov}}(X, Y)}{\widehat{\text{var}}(X)}, \quad \text{and} \quad \hat{a} = \bar{Y} - \hat{b}\bar{X}.$$

We interpret a as the mean value of Y when $x = 0$, and it is called the *intercept* (with the y-axis). The regression coefficient b specifies how much the expected value for Y grows for each unit x has grown, and it is called the *slope*. The conditional expectation for Y_i is then predicted as

$$\hat{Y}_i = \hat{a} + \hat{b}X_i.$$

$\hat{Y}$ is called the predicted value. The difference between observed value and the predicted value

$$r_i = \hat{Y}_i - Y_i$$

is an estimate of the ε_i and called residuals. The variance σ^2 of ε_i can be estimated by the mean-squared residuals

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n} r_i^2$$

where the denominator is corrected for the two degrees of freedom lost in estimating the two parameters a and b.

5.1.1.2 The Multivariate Case

In the case that we want to use more variables, e.g., predict $D = 3$ variables, in a row-vector $\mathbf{Y}$, as linear functions of $P = 2$ explanatory variables, in a row-vector $\mathbf{X}$, the linear regression model becomes

$$\begin{aligned}\mathbf{Y} &= [Y_1, Y_2, Y_3] = [a_1, a_2, a_3] + [X_1, X_2] \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} + [\varepsilon_1, \varepsilon_2, \varepsilon_3] \\ &= \mathbf{a} + \mathbf{X} \cdot \mathbf{B} + \varepsilon \end{aligned} \tag{5.1}$$

which can be seen at the same time as a linear combination of random coefficients with the rows of $\mathbf{B}$,

$$\mathbf{Y} = \mathbf{a} + X_1 \cdot \mathbf{b}_{1\cdot} + X_2 \cdot \mathbf{b}_{2\cdot} + \varepsilon, \tag{5.2}$$

or as three separate scalar products of the columns of **B** with a random vector,

$$Y_i = a_i + [X_1, X_2] \cdot [b_{1i}, b_{2i}]^t + \varepsilon_i = a_i + \mathbf{X} \cdot \mathbf{b}_{\cdot i}^t + \varepsilon_i, \tag{5.3}$$

one for each coordinate of **Y**. The coefficient estimators are essentially the same as in the univariate case, namely,

$$\hat{\mathbf{B}} = (\widehat{\text{var}}[\mathbf{X}])^{-1} \cdot \widehat{\text{cov}}[\mathbf{X}, \mathbf{Y}], \quad \text{and} \quad \hat{\mathbf{a}} = \bar{\mathbf{Y}} - \bar{\mathbf{X}} \cdot \hat{\mathbf{B}},$$

and for the errors, we find a residual covariance matrix of

$$\hat{\Sigma} = \frac{N-1}{N-P}(\widehat{\text{var}}[\mathbf{Y}] - \widehat{\text{cov}}[\mathbf{Y}, \mathbf{X}] \cdot \hat{\mathbf{B}}),$$

where the fraction just corrects for the actual number of estimated parameters.

5.1.1.3 The R Interface

In **R**, using for illustration a built-in demo dataset about orange trees, the linear model is accessed through the command `lm`:

```
> data(Orange)
> names(Orange)

[1] "Tree"          "age"           "circumference"

> model = lm(circumference~age,data=Orange)
> model

Call:
lm(formula = circumference ~ age, data = Orange)

Coefficients:
(Intercept)          age
     17.400        0.107
```

This command generates an object encapsulating the estimated model, on which various further tasks can be performed with accessory commands. For instance, the resulting line $\hat{y}(x) = \hat{a} + \hat{b}x$ can be drawn into scatterplots (command `abline`) or used to compute predicted values $\hat{y}_i = \hat{a} + \hat{b}x_i$ (command `predict`) and residual values computed as $r_i = y_i - \hat{y}_i$ (command `resid`).

```
> plot(circumference~age,data=Orange)
> abline(model)
> points(predict(model)~age,data=Orange,pch=20)
```

The following two commands each compute the estimated variance $\hat{\sigma}^2$:

```
> var(model)

[1] 563.5

> summary(model)$sigma^2

[1] 563.5
```

To judge the probable true range of the parameters, we would compute a confidence limit based on t-distributions:

```
> confint(model)

                2.5 %  97.5 %
(Intercept) -0.14328 34.9426
age          0.08993  0.1236
```

Likewise, we would be interested in confidence intervals for the predictor (for the average of Y as a function of x) and prediction intervals (for the actual values of Y as a function of x). That is again provided by the generic command `predict`

```
> mynew = seq(from=min(Orange$age),to=max(Orange$age),
                                                  length.out=100)
> mynew = data.frame(age=mynew)
> options(max.print=40)
> (CI = predict(model,newdata=mynew, interval="confidence"))

       fit    lwr    upr
1    30.00  14.19  45.81
2    31.58  15.98  47.18
3    33.16  17.77  48.54
4    34.74  19.56  49.91
5    36.31  21.35  51.28
6    37.89  23.13  52.65
7    39.47  24.92  54.02
8    41.05  26.70  55.40
9    42.63  28.49  56.77
10   44.21  30.27  58.15
11   45.79  32.05  59.53
12   47.37  33.83  60.91
13   48.95  35.60  62.29
 [ reached getOption("max.print") -- omitted 87 rows ]]

> (PI = predict(model,newdata=mynew, interval="prediction"))

       fit      lwr    upr
1    30.00 -20.8185  80.82
```

```
2    31.58 -19.1738  82.33
3    33.16 -17.5302  83.84
4    34.74 -15.8878  85.36
5    36.31 -14.2465  86.87
6    37.89 -12.6064  88.39
7    39.47 -10.9675  89.91
8    41.05  -9.3297  91.43
9    42.63  -7.6930  92.95
10   44.21  -6.0576  94.47
11   45.79  -4.4233  96.00
12   47.37  -2.7901  97.52
13   48.95  -1.1582  99.05
 [ reached getOption("max.print") -- omitted 87 rows ]]
```

```
> options(max.print=99999)
```

The first two lines of this example build a dataset with the new values of the explanatory variable for which we want the confidence or prediction intervals, in fact a dense sequence between the maximum and the minimum observed values of tree age in the dataset. The following two lines do the actual computations, now returning data frames with three columns (the prediction, the lower interval limit, and the upper interval limit). Adding them to the plot is quickly done by the command

```
> matlines(mynew, CI, col="red", lty=2)
> matlines(mynew, PI, col="black", lty=3)
```

where the accessory parameters `col` and `lty`, respectively, control the color of lines and the line style (in these two cases, dashed and dotted). The `matlines(x,y)` plots the columns of `y` against those of `x` as lines.

5.1.1.4 Checking the Model Quality

We need to do a test checking whether there is enough evidence of dependence. This is typically done through an ANOVA (analysis of variance) table:

```
> anova(model)
```

```
Analysis of Variance Table

Response: circumference
          Df Sum Sq Mean Sq F value  Pr(>F)
age        1  93772   93772     166 1.9e-14
Residuals 33  18595     563
```

In general multivariate regression, ANOVA tables have one line for every independent variable in the model, plus one line for the residuals. Each of the lines for the independent variables contains some information regarding a test of the form:

- H_0, null hypothesis: the slope coefficient b_i linked to the variable of the ith is zero; thus, that independent variable x_i does not apport any significant contribution to the prediction of Y.
- H_1, alternative hypothesis: the independent variable of the line is additionally needed, i.e., its coefficient $b_i \neq 0$.

In our univariate regression case, we only have one independent variable, and thus the test reads

$$H_0 : b = 0 \text{ vs. } H_1 : b \neq 0$$

where a model of the form $Y_i = a + 0X_i + \varepsilon_i$ would describe the fact that X does not have a linear influence on Y.

The p-value of the corresponding test is found in the `Pr(>F)` column of the analysis of variance table. In this case, a p-value of $1.9e-14$ is considerably smaller than the usual critical α-level, such as say $\alpha = 0.05$. Thus, the null hypothesis is rejected and—still subject to some check of assumptions—thus proven that oranges of older trees tend to be larger in statistical population sampled. That they are greater, and not smaller, can be seen from the positive sign of the estimated parameter, $\hat{b} > 0$.

We would also use a correlation coefficient or R^2 as a measure of how important the age is for determination of the circumference:

```
> summary(model)$r.squared

[1] 0.8345

> R <- with(Orange,cor(circumference,age))
> R^2

[1] 0.8345
```

A good regression analysis should further include a thorough analysis of the residuals and be eventually based on a robust estimation procedure. These issues will be directly treated in the next sections, devoted to regression with compositional variables.

5.1.2 Classical Analysis of the Variance (Discrete Covariables)

5.1.2.1 The Simple ANOVA Model

To analyze the dependence of a real random variable Z on a categorical covariable X (in **R** terminology, a *factor*), with m possible outcomes $\{g_1, \ldots, g_m\}$, we typically use an analysis of variance (ANOVA) model of the form

$$Z_i = a + b_{X_i} + \varepsilon_i$$

where a and the $b_g, g \in \{g_1, \ldots, g_m\}$ are real constants. The residual term ε_i is again a zero-mean random variable with constant variance, modeled with a normal distribution. Analysis of the variance is also used where the sample can be split in several subsamples, and we want to check whether the several subsample means differ significantly.

In an ANOVA model, each of the possible levels of the factor has a different conditional expected value:

$$\mu_g := E[Z|X = g] = a + b_g$$

Thus, each b_g describes the expected value of X in a subsample or for each level of the factor $g \in \{g_1, \ldots, g_m\}$. However, this model specification has a flaw: it is not unique. By changing the constant a to a constant a', we can always replace b_g by $b'_g = b_g - (a' - a)$ and get the same deterministic part in the model equation:

$$a' + b'_g = a + (a' - a) + b_g - (a' - a) = a + b_g = \mu_g$$

It is thus not possible to distinguish between the set of constants a, $b_g, g \in \{g_1, \ldots, g_m\}$ and the modified set a', $b'_g, g \in \{g_1, \ldots, g_m\}$. In this case, one typically says that the parameters were *non-identifiable*. To avoid this problem, the software modifies the specification into an equivalent model with identifiable parameters. These identifiable parameters are called *contrasts* and are linear combinations of the μ_g.[1]

5.1.2.2 Working with Factors

R is often able to identify categorical variables and automatically gives them the class "`factor`". A factor is in fact just a vector of integer numbers, with some complementary attributes: `levels`, `ordered`, and eventually `contrasts`.

The attribute `levels` is an ordered list of the possible outcomes of this factor, also called *categories*. Because the factor only stores numbers, this list is necessary to recode each number to its actual value. For instance, if a factor has levels "Africa," "America," and "Asia," a 1 in the factor list is taken as "Africa," a 2 as America and so on. If we need to change the whole set of levels (for instance, to abbreviate or correct it), we can use the command `levels`

```
> levels(X) <- c("Afr","Ame","Asi")
```

We can also force a factor to forget its levels and work as an integer vector by the command `as.integer` and, vice versa, force a vector of integers to a factor by

[1] In a strict sense of contrasts, the weights need to have sum 0. However, **R** does not always obey to this restriction.

`as.factor`. However, in that case the factor levels will be just numbers, and we will need to use `levels` again to redefine the lost names.

An alternative way to define a factor is to use the command `factor`

```
> X = factor(datasource, levels=
             c("never","often","always"), ordered=TRUE)
```

where `datasource` is the vector of observations, `levels` is the list of categories, and `ordered` (default `FALSE`) tells **R** whether the ordering given to levels is meaningful, or not. Note that this last argument becomes directly an extra attribute called `ordered`.

The `table` command allows to count how often each of the categories shows up in the dataset. The same result is obtained when applying `summary` to a factor

```
> table(X)
```

Finally, we can also store in the factor the contrast we want to use in the future. The actual model contrasts used for parameter estimation can be controlled by the commands `contrasts` or `options(contrasts=)`. For unordered factors, the default corresponds to the command `contr.treatment` and simply forces $b_{g_0} = 0$ such that

$$a = E[Z|X = g_0]$$
$$b_g = E[Z|X = g] - E[Z|X = g_0]$$

The code in this book will rely on that default. For ordered factors, the default corresponds to the command `contr.poly` and uses a more complicated description of the same model. We will avoid that complication by explicitly setting the contrasts:

```
> contrasts(X)<-"contr.treatment"
```

5.1.3 The Different Types of Compositional Regression and ANOVA

Compositional regression is quite analogous to multiple linear regression, although more complex in the details. For each method, we need to

- Generalize the mathematical definitions
- Learn how to perform the analyses and tests in **R**
- Display the model (parameters, predictions, residuals, intervals, etc.) in a way respecting the compositional nature of some of the involved variables
- Understand how to interpret our results (statistics and tests) as compositions

We can distinguish up to four types of compositional regression: Y could be a compositional variable (with continuous ordiscrete covariables, i.e., with regression

or ANOVA), or X could be a compositional variable, or both X and Y could be compositional. Finally, special attention should also be paid to the situation where X and Y are different aspects of the same composition.

5.2 Compositions as Independent Variables

5.2.1 Example

Let us consider a toy example. We are designing a new specialized potting substrate for strawberries. Within certain limits, we can control the mineral composition of the substrate. We would like to understand how the substrate influences the total biomass of strawberry produced per season. We conduct an experiment where we measure the biomass in kg of strawberries from a defined pot size filled with different substrates of designed composition.

```
> Strawberries = read.csv("Strawberries.csv")
> Strawberries
```

```
      C1     C2     C3 biomass
1 0.3333 0.3333 0.3333  12.406
2 0.1667 0.6667 0.1667   6.004
3 0.6667 0.1667 0.1667   4.389
4 0.1667 0.1667 0.6667  13.710
5 0.4444 0.4444 0.1111   4.939
6 0.1111 0.4444 0.4444  11.653
7 0.4444 0.1111 0.4444  11.267
```

```
> X            = acomp( Strawberries[,1:3] )
> biomass      = Strawberries$biomass
```

Since biomass has a ratio scale and can only be positive, we could consider a log transformation. As a justification, note that with unlimited water and light, plant biomass production might be considered as exponential. We will thus use $y = \ln(\text{biomass})$ as the dependent variable.

```
> Y = log(biomass)
```

5.2.2 Direct Visualization of the Dependence

The most direct, intuitive visualizations of dependence plot the scatter of the variable of interest in the Y-axis against another one deemed as explanatory, in X-axis. That is in the case of compositional explanatory data not so easy, due to the inherently multivariate nature of what we plan to plot in the X-axis. We would need

several dimensions to represent the scatter of our explanatory variable, before being able to show how it influences the scatter of the explained one. To overcome that problem of dimensionality, we can use several tools:

- We can plot a compositional representation of $\mathbf{X}$ and use color or symbol size as a scale to represent Y.
- We can represent Y against all pairwise log ratios of $\mathbf{X}$, in the fashion of the variation matrix (Sect. 4.1.3) or log-ratio boxplots (Fig. 4.3).
- We can represent Y against some balances or some coordinates of $\mathbf{X}$, either defined a priori or coming from some previous analyses.

The last option may be interesting when the basis used to represent $\mathbf{X}$ is somehow interpretable. The other two options are more exploratory and are presented in detail in the next subsections.

5.2.2.1 Using Symbol Size and Color

The idea behind this technique is to establish a (quasi-)continuous scale of size or a color ramp, store this information in a vector of the same size as Y, and pass it as an extra argument to a ternary diagram:

```
> rescale = function(xin,rout,rin=range(xin, na.rm=FALSE)){
+   xout = rout[1] + (xin-rin[1])*(rout[2]-rout[1])/(rin[2]-rin[1])
+   return(xout)
+  }
> Ycex = rescale(Y, c(0.5,2) )
> Ycol = grey( rescale(Y,c(0.1,0.8)) )
> plot(X,cex=Ycex)
> plot(X,pch=19,col=Ycol)
```

In this code chunk, we have defined a function that linearly rescales an input vector `xin` so that the range `rin` becomes the range `rout`. That function is afterwards used to get a scale of radius for the circles and of grey levels for the colors. The function `grey` takes a vector of values between 0 and 1 and returns the corresponding grey level (0 is black; 1 is white). There is no function in the **R** basic distribution to obtain color ramps (e.g., gradually passing from red to white to blue or from yellow to magenta to blue). That must be obtained using function `mixcolor` from library "`colorspace`". The following code, for instance, generates a ramp from red to blue:

```
> require("colorspace")
> Ycol = hex(  mixcolor( rescale(Y,c(0,1)), RGB(1,0,0),
                                            RGB(0,0,1)) )
```

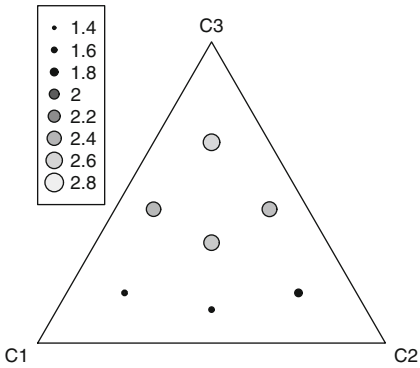


Fig. 5.1 Direct visualization of the influence of a dependent variable (log-transformed biomass) on a composition (of the substrate), captured as color filling and size of the symbols

Plots with colored and increasing size symbols are meaningless without a legend. The following code generates a plot with combined symbol size and gray-scale colors and adds the adequate legend (Fig. 5.1). The function `pretty` generates a vector of "nice" values to serve as scale for `Y`:

```
> opar <- par(mar=c(3,3,0,0))
> plot(X,pch=21,bg=Ycol,cex=Ycex)
> (Ylev <- pretty(Y))

[1] 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8

> legend(x=0,y=0.97,pch=21,legend=Ylev,
+   pt.cex=rescale(Ylev, c(0.5,2),range(Y)),
+   pt.bg=grey(rescale(Ylev,c(0.1,0.8),range(Y)) )
+   )
> par(opar)
```

5.2.2.2 The Pairwise Log-Ratio Plot Matrix

The second option is to extensively explore how the target variable Y depends on the simplest set of projections available, all possible pairwise log ratios (Fig. 5.2). In the current version, there is no packed function doing this job, but the result can be quickly obtained by nesting two loops:

```
> opar <- par(mfrow=c(3,3),mar=c(1,1,0.5,0.5), oma=c(3,3,0,0))
> for(i in 1:3){
+  for(j in 1:3){
+   plot(log(X[,i]/X[,j]),Y,pch=ifelse(i!=j,19,""))
+   if(i==j){text(x=0,y=mean(range(Y)),
+             labels=colnames(X)[i],cex=1.5)}
+  }
+ }
```

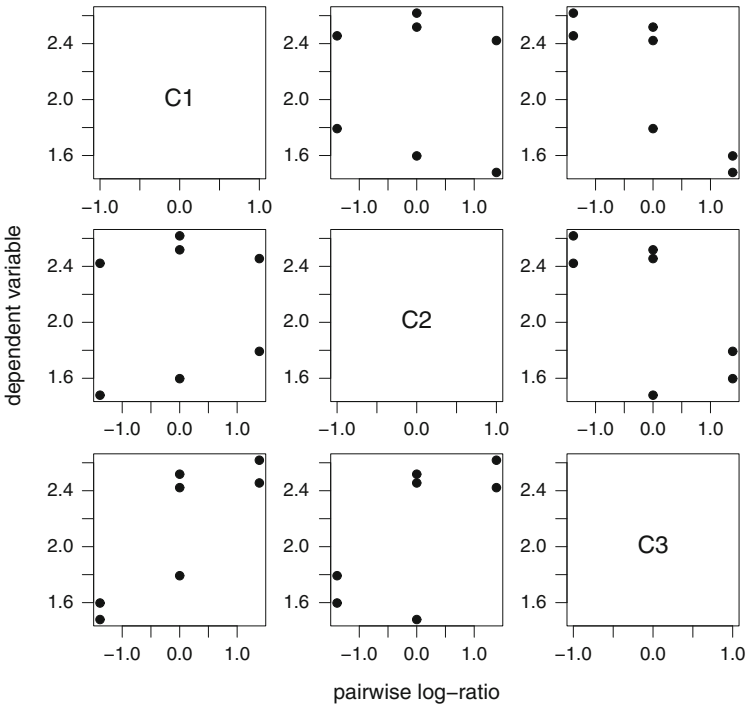


Fig. 5.2 Dependent variable plotted against all possible pairwise log ratios of the explanatory composition: the X-axis of each diagram represents the log ratio of the variable in the row against the variable in the column

```
> mtext(text=c("pairwise log-ratio","dependent variable"),
                    side=c(1,2),at=0.5,line=2,outer=TRUE)
> par(opar)
```

Two tricks are used here to get appealing results. By giving to the plotting function the accessory argument `pch=ifelse(i!=j,19,"")`, we plot the data as dots (`pch=19`) in the off-diagonal diagrams (`i!=j`, $i \neq j$) and invisible (`pch=""`) in the diagonal diagrams. The second trick is the automatic placement of the variable names of **X** as labels in the diagonal elements in the middle of their boxes (because the log ratio of a variable against itself is identically zero, and the middle of the vertical axis is the center of the range of the Y variable). Function `mtext` just places explanatory labels on the margins of the figure. From version 1.30 of the package, the function pwlrPlot does this job automatically.

5.2.3 The Model

There are multiple ways to generalize a linear regression to a composition. Compositions are per definition multivariate and thus must be mapped somehow,

linearly or nonlinearly to a single number. In parallelism with (5.3), that mapping would most conveniently be done for compositions by an Aitchison scalar product (2.4), leading to a regression model of the form

$$Y_i = a + \langle \mathbf{b}, \mathbf{X}_i \rangle_A + \varepsilon_i$$

with a real random variable Y and an independent compositional variable $\mathbf{X}$ and where the parameters are a real-valued intercept a, a regression "slope" composition $\mathbf{b} \in \mathbb{S}^D$, and a real-valued zero-mean error ε, typically modeled as normally distributed.

The composition $\mathbf{b}$ has a simple interpretation. It gives the direction along which $\mathbf{X}$ must be perturbed to achieve the largest effects on Y. If two observations $\mathbf{x}_i$ and $\mathbf{x}_j$ differ by a unit vector $\frac{\mathbf{b}}{\|\mathbf{b}\|}$ in the direction of $\mathbf{b}$, i.e., $\mathbf{x}_j = \mathbf{x}_i \oplus \frac{\mathbf{b}}{\|\mathbf{b}\|}$, then the expected value of Y_j differs from the expect value of Y_i by an amount of $\|\mathbf{b}\|$:

$$\begin{aligned} E[Y_j | X_j = x_j] &= a + \langle \mathbf{b}, \mathbf{x}_j \rangle_A + \overbrace{E[\varepsilon]}^{0} = a + \left\langle \mathbf{b}, \mathbf{x}_i \oplus \frac{\mathbf{b}}{\|\mathbf{b}\|} \right\rangle_A \\ &= \underbrace{a + \langle \mathbf{b}, \mathbf{x}_i \rangle_A}_{E[Y_i]} + \frac{1}{\|\mathbf{b}\|} \underbrace{\langle \mathbf{b}, \mathbf{b} \rangle_A}_{\|\mathbf{b}\|^2} = E[Y_i | X_i = x_i] + \|\mathbf{b}\| \end{aligned}$$

5.2.4 *Estimation of Regression Parameters*

To compute such a regression model in **R**, we would use the principle of working in coordinates to transform the model into a multiple regression problem:

$$\begin{aligned} Y_i &= a + \langle \mathbf{b}, \mathbf{X}_i \rangle_A + \varepsilon_i \\ Y_i &= a + (\mathrm{ilr}(\mathbf{b}), \mathrm{ilr}(\mathbf{X}_i)) + \varepsilon_i \\ Y_i &= a + \sum_{k=1}^{D-1} \mathrm{ilr}(b)_k \mathrm{ilr}_k(X_i) + \varepsilon_i \\ Y_i &= a + \sum_{k=1}^{D-1} \beta_k \mathrm{ilr}_k(X_i) + \varepsilon_i \end{aligned}$$

with a vector of slope parameters $\beta = (\beta_k)$ that later might by mapped back to a composition through the inverse ilr transformation ilr^{-1}:

```
> (model = lm(Y~ilr(X)))

Call:
lm(formula = Y ~ ilr(X))
```

```
Coefficients:
(Intercept)      ilr(X)1      ilr(X)2
     2.1261       0.0885       0.5371
```

```
> (a = coef(model)[1])
```

```
(Intercept)
      2.126
```

```
> (b = ilrInv(coef(model)[-1],orig=X))
```

```
    C1     C2     C3
0.2387 0.2706 0.4907
attr(,"class")
[1] acomp
```

Remark 5.1 (Linear models with other log-ratio transformations). For most of the operations in the chapter, we could use the alr transformation with the same right as the ilr transformation. The isometry of the operation is only relevant when metric variances are compared. This happens, for instance, when measures of determination like R^2 are generalized: in these situation, only the ilr transformation fulfills the requirement that the analysis is permutation invariant, because the alr transformation changes the geometry. It is thus safest to always use the ilr transformation in this chapter. It has been argued that results are more interpretable in alr than in ilr: this is not so important in our approach, because we will interpret the back-transformed results as compositions. The clr transformation is not easy to use in the context of linear models with compositional explanatory variables, because it introduces numerical problems with singular matrices in tests.

5.2.5 Displaying the Model

5.2.5.1 The Model as a Surface

The easiest representation of the model is to plot a line along the estimated slope parameter $\hat{\mathbf{b}}$ passing through the center of our dataset:

```
> plot(X)
> straight(mean(X), b, lwd = 2, col = "black",lty=2)
```

We can also place some dots as landmarks along this line, at regular intervals of the predicted value:

```
> myY = pretty(Y)
> refX = mean(X) + ((myY - a)/norm(b)^2) * b
> plot(refX, add = TRUE, pch = 19)
```

The estimated regression model can be visualized as a surface on this ternary diagram, i.e., $Y = f(\mathbf{X})$, a surface increasing only in the direction of $\hat{\mathbf{b}}$. In fact,

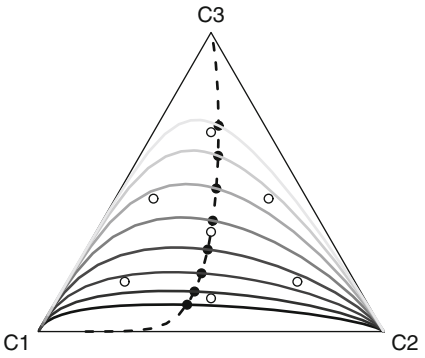


Fig. 5.3 Surface of the fitted model (in *gray scale*), with maximum-gradient direction and some fixed $\hat{Y}$ value landmarks

in the orthogonal direction to $\hat{\mathbf{b}}$, the predictions $\hat{Y}$ happen to be constant. It is thus easy to plot this surface by representing a family of lines (Fig. 5.3):

```
> orthoComp = function(x) { ilrInv(ilr(x) %*%
                        matrix(c(0, 1, -1, 0), ncol = 2)) }
> mygreyscale = grey((0:nrow(refX))/nrow(refX))
> for (i in 1:nrow(refX)) {
+   straight(acomp(refX[i, ]), orthoComp(b),
                             col = mygreyscale[i],lwd = 2)
+ }
```

The first line of this code defines an ad hoc function to compute the orthogonal composition of a 3-part composition, by the trick of permuting the two ilr coordinates and changing the sign of one of them. The second command just defines a regular color scale of grays, by giving a series of numbers between 0 and 1. This is a built-in function. The loop is finally plotting a line for each row in `refX`, the reference points computed before. It would still be necessary to add labels to each line, showing the associated $\hat{Y}$ values. It would be also possible to get this figure adapting the code chunk in Example 3.2.

5.2.5.2 Parameter Confidence Regions

The estimation variance of **b** as a composition is given by

```
> (varb=ilrvar2clr(vcov(model)[-1,-1]))
```

```
           1          2          3
1  0.011389 -0.005694 -0.005694
2 -0.005694  0.011389 -0.005694
3 -0.005694 -0.005694  0.011389
```

which can be plotted as a confidence ellipsoid (Fig. 5.4). Assuming joint normality of all our variables (the composition expressed in ilr), we can compute the size of this ellipse with a Fisher F distribution by the following commands:

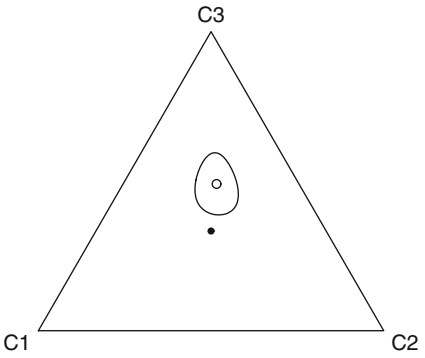


Fig. 5.4 Confidence region for a compositional parameter. The *circle* plotted represents the compositional parameter estimate, and the *dot* is the neutral element of the simplex

```
> par(mar=c(3,3,0,1))
> names(b) = colnames(X)
> scaleB   = 1
> plot(scaleB*b)
> plot(0*b,add=TRUE,pch=20)
> alpha = 0.05
> rF = sqrt(qf(1-alpha,nrow(varb)-1,model$df))
> ellipses(scaleB*b,scaleB^2*varb,rF)
```

The numerator degrees of freedom result from the number of dimensions along which the parameter can vary. The denominator degrees of freedom results from the residual degrees of freedom in the model, determining the precision of estimation of the residual variance and thus of the estimator covariance provided by command `vcov`. The parameter `scaleB` allows to report a rescaled version of the figure, when the ellipse is too small (give a larger value) or when the dot is plotted too near to the borders (give a smaller value).

This figure is quite easy to interpret. The central dot (the zero or neutral element of the simplex) represents a vector which would induce no change in the response Y. If the confidence ellipse comprises this dot, we should accept that the estimated parameter is not significant enough to distinguish it from the zero, and thus there is no dependence of the explained variable on the composition. On the contrary, if the central dot falls outside the confidence ellipse like in this example, we can assume with a $(1-\alpha)$ confidence that there is a significant dependence of log-biomass with the substrate composition.

5.2.6 Prediction and Predictive Regions

To predict the value for another new composition X, we can simply write a `newdata=` argument into the `predict` command enclosing the new value for X in a named list:

```
> newX <- acomp(c(1,2,3))
> (newPredicted <- predict(model,newdata=list(X=newX)))

    1
2.499
```

We can also use the same predict command to generate confidence and probability intervals, by giving the accessory argument `interval="confidence"`, respectively, `interval="prediction"`, as well as the desired confidence `level`

```
> predict(model, newdata = list(X = newX),
                    interval = "confidence", level = 0.95)

    fit   lwr   upr
1 2.499 2.107 2.891
```

It is far more difficult to get a proper predictive interval in that situation, which accounts for all uncertainties, adding up variability from prediction and confidence:

```
> (newDesignmatrix = model.matrix(terms(model),
                                  data=list(X=newX,Y=0)))

  (Intercept) ilr(X)1 ilr(X)2
1           1  0.4901   0.614
attr(,"assign")
[1] 0 1 1

> (errVar  = newDesignmatrix %*% vcov(model)%*% t(newDesignmatrix))

        1
1 0.01992

> alpha=0.05
> (confidenceInterval = newPredicted +
   qt(c(alpha/2,1-alpha/2),model$df.residual)*sqrt(errVar))

[1] 2.107 2.891

> (predictiveInterval = newPredicted + qt(c(alpha/2,1-alpha/2),
              model$df)*sqrt(summary(model)$sigma^2+errVar))

[1] 1.687 3.312
```

Here `confidenceInterval` is again the $1-\alpha$ confidence interval for the conditional expectation of Y given $X =$ `newX` assuming the model to be true. On the contrary, `predictiveInterval` is the $1-\alpha$ predictive interval for the random Y observed for a $X =$ `newX`, gathering the uncertainty on the estimation of all parameters of the model.

5.2.7 Model Checks

As a matter of fact, displaying and predicting with a regression model should only be done after ensuring a minimal quality of the model. The next sections cover the most typical graphical and numerical checks:

- Assessing the strength of the relationship (Sect. 5.2.8, page 112), comparing the observed explained values against their predictions by the model or by a Fisher F test checking that reduction of the uncertainty about the explained variable is substantial enough.
- Checking that some coefficients cannot be removed (Sect. 5.2.9, page 114), simplifying the model (i.e., that a given subcomposition plays no role in predicting the explained variable).
- Checking the absence of leverage points and other similar bad-behaved data that could influence too much the model (Sect. 5.2.10, page 115).
- Checking the assumption that residuals are normally distribution (Sect. 5.2.11, page 117).
- Checking that their variance is constant (Sect. 5.2.12, page 118).

If some of these checks are not satisfactory and especially if there are influential or leverage points, a robust regression procedure should be used instead (Sect. 5.2.13, page 118).

5.2.8 The Strength of the Relationship

Unlike in univariate linear regression, it is not possible to display the strength of the relationship between several compositional variables and a dependent variable in a single XY scatterplot, because $\mathbf{X}$ has several components potentially influential. We can nevertheless replace the regressor $\mathbf{X}$ by the optimal predictor $\hat{Y}$ (see Fig. 5.5):

```
> opar <- par(mar=c(3,3,0,0))
> Predicted    = predict(model)
> plot(Predicted,Y)
> abline(0,1)
> par(opar)
```

`predict` computes the predicted values $\hat{Y}_i := \hat{a} + \left\langle \hat{\mathbf{b}}, \mathbf{X} \right\rangle_A$ of the model. `abline(0,1)` draws the bisecting line of the first quadrant, $Y = 0 + 1\hat{Y}$, representing a perfect agreement of predicted and observed values.

Two measures of determination R^2 and R^2_{adj} can be used to numerically evaluate the strength of the relationship. The theoretical R^2_{theo} is given by

$$R^2_{\text{theo}} = \frac{\text{var}(\hat{Y})}{\text{var}(Y)} = 1 - \frac{\text{var(residuals)}}{\text{var}(Y)}$$

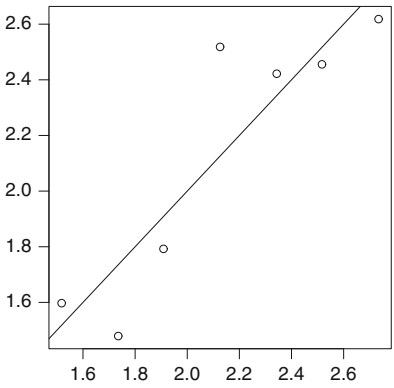


Fig. 5.5 A scatterplot of predicted vs. observed values can replace some aspects of the classical dependent vs. independent scatterplot if compositions are used as independent variables in a regression

and interpreted as the portion of the total variability var(Y) that is explained by the model. As a portion, it is strictly between 0 and 1, zero meaning no dependence and one corresponding to a perfect explanation of the behavior of Y through the knowledge of X without any need of residual terms. The theoretical R^2_{theo} is often estimated by the empirical R^2:

$$R^2 = \frac{(1/n)\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{(1/n)\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

However, the empirical R^2 overestimates R^2_{theo}, and the bias increases with increasing number of parameters. This could lead to an unjustified preference of models with more parameters. A bias-adjusted version

$$R^2_{\text{adj}} = 1 - \frac{(1/(n-df))\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{(1/(n-1))\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

estimates these variances with corrected degrees of freedom and yields an unbiased estimate, but it can lead to negative values and is therefore slightly more difficult to interpret.

Another difference with linear regression: it is no longer possible to evaluate the strength of the relationship with a true correlation coefficient, since there is no clear "upward" direction in a composition that could help to discern negative and positive correlation. For those more familiar with correlations than with R^2, we can identify $\sqrt{R^2}$ with the absolute value of the Pearson correlation coefficient, because it is mathematically equivalent in all cases where the Pearson correlation is defined. As a reference, one might compute the corresponding Pearson correlation, which is numerically the same as the square root of R^2:

```
> cor(Y,Predicted,method="pearson")

[1] 0.9015
```

A good way to display the relationship of variance originating from prediction and residuals is also a centered parallel boxplot of all three variables (not shown):

```
> Residuals    = model$residuals
> boxplot(list(Y-mean(Y),Predicted-mean(Y),Residuals))
```

5.2.9 Global and Individual Tests

The model summary and the table of analysis of variance are providing several tests, as well as both measures of determination, R^2 and R^2_{adj}:

```
> summary(model)

Call:
lm(formula = Y ~ ilr(X))

Residuals:
      1       2       3       4       5       6       7
 0.3921 -0.1165 -0.2563 -0.1159  0.0791 -0.0612  0.0786

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.1261     0.0969   21.95  2.5e-05
ilr(X)1       0.0885     0.1307    0.68    0.535
ilr(X)2       0.5371     0.1307    4.11    0.015

Residual standard error: 0.256 on 4 degrees of freedom
Multiple R-squared: 0.813,  Adjusted R-squared: 0.719
F-statistic: 8.67 on 2 and 4 DF,  p-value: 0.0351

> anova(model)

Analysis of Variance Table

Response: Y
          Df Sum Sq Mean Sq F value Pr(>F)
ilr(X)     2  1.139   0.570    8.67  0.035
Residuals  4  0.263   0.066
```

The ANOVA table provides an overall test for the complete parameter set. In this case, it suggests that the substrate composition somehow influences the biomass. On the contrary, the summary provides an individual test for the different coordinates of the ilr-transformed composition. In this example, the results suggest that only one ilr coordinate might be relevant for biomass production. However, if we use the standard ilr transformation, the individual ilr coordinates per se have seldom a direct interpretation, and their associated tests should therefore be ignored.

The case is different if we want to test the role of a particular subcomposition. Take a composition and split it in two subcompositions, A and B. Assume we want to test that the parts in the subcomposition A play no role. In this situation, the ilr basis should be chosen accordingly, as a balance basis (Sect. 4.3) separating A from B. We will have then some coordinates computed only with parts in A, some coordinates involving parts in B, and a coordinate balancing the components in subcomposition A against those components in the subcomposition B. If we accept all tests of the coefficients in the first group, we accept that relative changes within these parts play no role in the response, and we can remove that subcomposition from the model.

5.2.10 Model Diagnostic Plots

A thorough analysis of the behavior of regression residuals is necessary to ensure that the estimated model is credible and therefore interpretable and further useful for prediction.

A standard tool to display the residuals of a multivariate regression model is the predicted vs. residuals scatterplot. Since the model is not compositional in Y, the basic functions `predict` and `resid` can be used as usual to compute predicted and residual values:

```
> Predicted = predict(model)
> Residuals = resid(model)
> plot(Predicted,Residuals)
```

It shows in a single plot how the residuals depend on the predicted values. At least the following three diagnostic aspects should be considered to decide that residuals are well behaved:

- *Is there an apparent structure in residuals, e.g., a curved dependency?*
  This would be a hint towards a nonlinear relationship.
- *Does the variability of the residuals depend on the predicted value?*
  If that is the case, we might suspect heteroscedasticity, i.e., a different variance for different ε_i, which violates the underlying hypotheses of regression.
- *Are there noticeable gaps between different groups of predicted values?*
  If the answer is positive, this hints to the existence of groups in the regressor values. It is then reasonable to suspect that the dependence of Y on $\mathbf{X}$ is just apparent, a surrogate of the existence of several subgroups.

This diagram may also be directly obtained by *plotting the model itself*:

```
> par(mfrow=c(2,2))
> plot(model,add.smooth=FALSE)
```

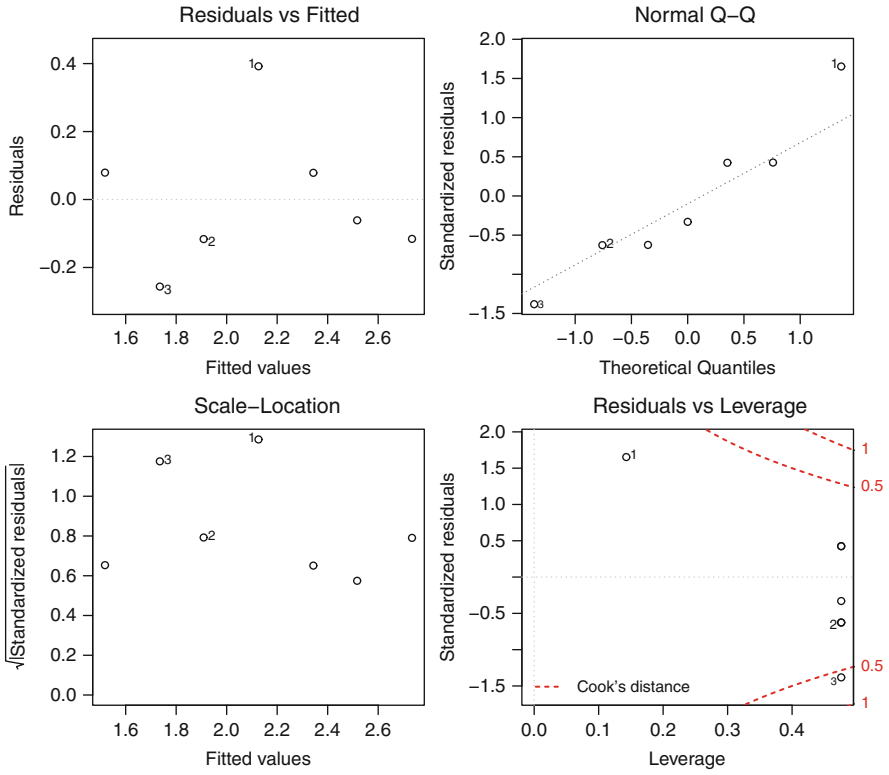


Fig. 5.6 Four standard diagnostic plots helping to detect outliers or violations of the hypothesis underlying the regression procedure

which generates four diagnostic plots (Fig. 5.6):

1. The already-mentioned residuals vs. predicted (or fitted values) plot
2. A normal QQ plot (explained in Sect. 5.2.11)
3. A scale-location plot (explained in Sect. 5.2.12)
4. A leverage plot

Leverage values Belsley et al. (1980) and Cook's distances (Cook and Weisberg, 1982) help in the identification of potential outliers in the model. These statistics can be obtained with the following basic **R** commands:

```
> Leverages = hatvalues(model)
> boxplot(Leverages)
> CooksDistances = cooks.distance(model)
> plot(Predicted,CooksDistances)
```

The leverage measures the potential influence of a given observation on its own predicted value. Its inverse, 1/leverages, can be interpreted as the equivalent number

of data in our dataset that support that given observation, e.g., a leverage of 0.25 would mean that four observations (including itself) are consistent with a given datum, i.e., that datum could be removed and quite well predicted from the other 3. On the contrary, a datum with a leverage of 1 would not be reproducible if we remove it from the dataset, i.e., there is no way to check the value of this observation. To evaluate leverages, we may use the following rule of thumb. First, we should ask ourselves which should be the minimum size of a dataset, so that a local estimation of the mean is reliable enough. Then, any datum with a leverage *larger* than the *inverse* of this minimal size should be considered suspicious, too influential. Leverages are computed only with the values of the explanatory variable $\mathbf{X}$; the dependent variable Y plays no role. Thus, leverages only provide information on the adequateness of the sampling design of $\mathbf{X}$ in the given dataset. Cook's distances, on the contrary, are measures of the actual influence of a given explained variable datum Y_i to its local prediction $\hat{Y}_i$. These two measures are therefore complementary: in the ideal case, all leverages should be smaller than our chosen threshold (one divided by the minimum trustable sample size), and the Cook's distances of all our observations should be within the envelope of 0.5 in the fourth diagnostic diagram.

5.2.11 Checking the Normality Assumptions

In a standard regression model with normally distributed errors ε_i, the computed residuals $\hat{\varepsilon}_i = y_i - \hat{y}_i$ are also normally distributed, but with different variances. We must then rescale the residuals to a common variance, a process called *studentization*. The studentized residuals can be reasonably assumed to be identically normally distributed. Though from the strictest point of view the studentized residuals are not independent (but typically slightly negatively correlated), we may still check normality and homoscedasticity with classical tools, ignoring that fact:

```
> StudentizedResiduals = rstudent(model)
> qqnorm(StudentizedResiduals)
> qqline(StudentizedResiduals)
> shapiro.test(StudentizedResiduals)

    Shapiro-Wilk normality test

data:  StudentizedResiduals
W = 0.9027, p-value = 0.3478

> boxplot(StudentizedResiduals)
```

The normal Quantile–Quantile plot, generated by the `qqnorm` command, suggests a normal distribution of the residuals if the points do not show a clear structure departing from a straight line. The Shapiro–Wilk test `shapiro.test` suggests non-normality if it shows a p-value below a critical level, e.g., $\alpha = 0.05$. The boxplot

suggests problems with the normality assumption if it shows outliers. QQ plots (with a normal reference QQ line) are generated also when plotting the model (Fig. 5.6).

5.2.12 Checking Constant Variance

Studentized residuals could also be used to detect departures from the assumption of *homoscedasticity*, i.e., the assumption that the errors ε_i have all the same variance. To check it, we need to use a variable potentially influencing the residual variance, like the predicted value. Then, we can test homoscedasticity checking the rank correlation of the absolute value of the studentized residuals against this suspected influence:

```
> cor.test(Predicted,abs(StudentizedResiduals),method="spearman")

    Spearman's rank correlation rho

data:  Predicted and abs(StudentizedResiduals)
S = 76, p-value = 0.4444
alternative hypothesis: true rho is not equal to 0
sample estimates:
    rho
-0.3571
```

The test would suggest *heteroscedasticity* (i.e., different error variances for different values of the independent variable) if the resulting p-value is lower than the critical level, e.g., $\alpha = 0.05$. For a visual inspection of that relation, it is a good idea to transform the residuals in a way reducing its variance, e.g.:

```
> plot(Predicted, abs(StudentizedResiduals))
> plot(Predicted, sqrt(abs(StudentizedResiduals)))
> plot(Predicted, log(abs(StudentizedResiduals)))
```

The second option is the one chosen by the default diagnostic plots (Fig. 5.6).

5.2.13 Robust Regression

Another possibility to study the presence of outliers and minimize its influence is to use robust estimation procedures, like those available from package "robustbase" (Rousseeuw et al., 2007). That entails no further difficulty, since we used a standard linear model for the actual computations.

```
> require(robustbase)
> (modelRob = lmrob(Y~ilr(X)))
```

```
Call:
lmrob(formula = Y ~ ilr(X))

Coefficients:
(Intercept)      ilr(X)1      ilr(X)2
     2.1199       0.0848       0.5350
```

```
> (aRob = coef(modelRob)[1])
```

```
(Intercept)
       2.12
```

```
> (bRob = ilrInv(coef(modelRob)[-1],orig=X))
```

```
    C1     C2     C3
0.2397 0.2702 0.4901
attr(,"class")
[1] acomp
```

Especially for robustly estimated models, it is often useful to check for extreme predicted values:

```
> ResidualsRob= resid(modelRob)
> PredictedRob= Y-resid(modelRob)
> plot(PredictedRob,Residuals)
```

Those might hint to extreme X values that might even not have been used to estimate the regression coefficient and would correspond to a wrong prediction.

In order to find possibly faulty values, not fitting into the total model, we should analyze the robust residuals for outlier, e.g., using a boxplot:

```
> bxpstats <- boxplot(ResidualsRob,plot=FALSE)
> plot(PredictedRob,Y,
       pch=ifelse(ResidualsRob %in% bxpstats$out,33,20))
```

Outliers in this boxplot hint to observation not fitting an overall regression model due to a robust estimation. The second plot (not shown) displays how these observations can be identified.

To evaluate the strength of the relationship, it might be reasonable to compute the analog of a rank correlation of the predicted values and the real values, instead of the classical coefficient of determination:

```
> cor(Y,Predicted,method="spearman")
```

```
[1] 0.8571
```

This approach would still rely on the linear model to detect the direction of the relationship and will therefore always result in a positive correlation coefficient. For the rest of checks on the properties of residuals, we could follow the preceding sections, as the hypotheses on the residuals are here, the same as in linear regression with compositional independent variable (normality, independence from predictions, homoscedasticity, absence of lever points, etc.).

5.2.14 Quadratic Regression

The compositional linear regression of the preceding section was driven by analogy to linear regression. However, a linear relationship might be too simple for our problem. For example, it is difficult to image that there is a way to tune a substrate composition towards an infinite yield. It seems more reasonable to assume the existence of a sort of optimum.

Obviously, we could linearize the dependence locally. However, it is reasonable to assume that when we are good at creating potting substrate, we will conduct experiments around the optimum value, and thus the best approximation would be a horizontal plane on that point. That is totally non-informative for our problem. Summarizing, we would need at least a quadratic model to correctly describe the local effect around the optimum, while a linear regression can be a good approximation far from it.

The second argument would be true if a component of the substrate is helping our plant but not essential. If it is an essential nutrient, it could well happen that the dependence is dramatic, i.e., killing the plant if it is nearly missing. If we do not believe in these extreme effects at the borders of the simplex, we might well use classical Euclidean geometry.

If we are interested in a quadratic dependence, we need a model of the form

$$Y_i = a + \langle \mathbf{b}, \mathbf{X}_i \rangle_A + \langle \mathbf{X}_i, C\,\mathbf{X}_i \rangle_A + \varepsilon_i$$

with a symmetric bilinear form C (i.e., a square symmetric matrix). Using the principle of working in coordinates, the model is translated to **R** as

$$\begin{aligned} Y_i &= a + \sum_j \mathrm{ilr}(\mathbf{b}_j)\mathrm{ilr}_j(\mathbf{X}_i) + \sum_{j,k} \mathrm{ilr}_j(\mathbf{X}_i)_j\, c_{jk}\, \mathrm{ilr}_k(\mathbf{X}_i) + \varepsilon_i \\ &= a + \sum_j \mathrm{ilr}(\mathbf{b}_j)\mathrm{ilr}_j(\mathbf{X}_i)_j + \sum_j c_{jj}\,\mathrm{ilr}_j(\mathbf{X}_i)_j + \sum_{j<k} c_{jk}\, 2\mathrm{ilr}_j(\mathbf{X}_i)\mathrm{ilr}_k(\mathbf{X}_i) \end{aligned}$$

```
> sq <- function(x) {
+     n <- if(length(dim(x))>1) ncol(x) else length(x)
+     j <- rep(1:n,n)
+     k <- rep(1:n,each=n)
+     s <- j<=k
+     f <- ifelse(j==k,1,2)[s]
+     if(length(dim(x))>1) f*x[,j[s]]*x[,k[s]] else x[j[s]]*x[k[s]]
+ }
> model2 = lm(Y~ilr(X)+sq(ilr(X)))
> model2

Call:
lm(formula = Y ~ ilr(X) + sq(ilr(X)))
```

```
Coefficients:
(Intercept)      ilr(X)1      ilr(X)2  sq(ilr(X))1  sq(ilr(X))2
      2.536        0.184        0.482       -0.308       -0.145
sq(ilr(X))3
     -0.327
```

```
> (a = coef(model2)[1])
```

```
(Intercept)
      2.536
```

```
> (b = coef(model2)[2:ncol(X)])
```

```
ilr(X)1 ilr(X)2
 0.1844  0.4817
```

```
> unsq <- function(x) {
+   n = (sqrt(8*length(x)+1)-1)/2
+   j <- rep(1:n,n)
+   k <- rep(1:n,each=n)
+   s <- j<=k
+   i <- j+(k-1)*n
+   C <- diag(n)
+   C[i[s]]<-x
+   C <- C+t(C)
+   diag(C) <- diag(C)/2
+   C
+ }
> (C = unsq(coef(model2)[-(1:ncol(X))]))
```

```
        [,1]    [,2]
[1,] -0.3080 -0.1445
[2,] -0.1445 -0.3272
```

Here $\mathbf{b}$ and C are expressed in ilr coordinates. We will denote this C expressed in ilr coordinates as $\mathbf{C}_V$, because it is a square symmetric matrix which values depend on the ilr matrix $\mathbf{V}$. If $\mathbf{C}_V$ is a definite matrix, we can then compute the ilr coordinates of the optimum of the estimated dependence by

$$\mathbf{0} = \nabla \hat{Y} = \nabla(a + \langle \mathbf{b}, \mathbf{X}_i \rangle_A + \langle \mathbf{X}_i, C\,\mathbf{X}_i \rangle_A) = \text{ilr}(\mathbf{b}) + 2\mathbf{C}_V \cdot \text{ilr}(\mathbf{x})$$

$$\text{ilr}(\mathbf{x}_{\text{opt}}) = \frac{1}{2}\mathbf{C}_V^{-1} \cdot \text{ilr}(\mathbf{b})$$

```
> (xOpt = ilrInv(solve(C,b)/2,orig=X))
```

```
    C1     C2     C3
0.4008 0.4350 0.1643
attr(,"class")
[1] acomp
```

Whether that optimum is a minimum, a maximum, or a saddle point depends on the eigenvalues of $\mathbf{C}_V$:

```
> eigen(C)

$values
[1] -0.1728 -0.4624

$vectors
        [,1]   [,2]
[1,] -0.7302 0.6833
[2,]  0.6833 0.7302
```

The optimum $\mathbf{x}_{\text{opt}}$ is a minimum if all eigenvalues are positive, it is a maximum if the eigenvalues are negative, and it is a saddle point if the eigenvalues do not have homogeneous signs. We are thus in this case finding a maximum.

However, as we can see from the analysis of variance table,

```
> anova(model2)

Analysis of Variance Table

Response: Y
           Df Sum Sq Mean Sq F value Pr(>F)
ilr(X)      2  1.139   0.570    68.3  0.085
sq(ilr(X))  3  0.254   0.085    10.2  0.226
Residuals   1  0.008   0.008
```

the quadratic part is not significantly different from zero. Thus, in this particular case, we should conclude that the quadratic behavior and the existence of a maximum is not sufficiently supported by the available data.

5.3 Compositions as Dependent Variables

5.3.1 Example

5.3.1.1 Continuous Covariables

The second type of compositional regression explains a composition from one or more real-scaled covariables. To illustrate this case, we will use a dataset of proportions of four petrographic types of sediment grains. These grain types are as follows: fragments of polymineralic rock (*Rf*), grains of one single quartz crystal (*Qm*), grains containing several quartz crystals (but only quartz, *Qp*), and grains of mica (*M*). The dataset was compiled by Grantham and Velbel (1988), who wanted to test how this composition depends on certain physiographic (relief ratio) and

climatic characteristics (river discharge) of the drainage basin where this sediment is generated.

```
> #require(compositions)
> GeoChemSed=read.csv("GraVel.csv",header=TRUE)
> Y=acomp(GeoChemSed[,7:10])
> names(GeoChemSed)
```

```
 [1] "watershed" "position"  "CCWI"      "discharge" "relief"
 [6] "grain"     "Qm"        "Qp"        "Rf"        "M"
```

```
> Covariables=GeoChemSed[,c("discharge","relief","grain",
                                                "position")]
> X1 = Covariables$discharge
> X2 = Covariables$relief
```

River discharge is a continuous covariable with a ratio scale, describing the outflowing annual volume of water per area of the drainage area above the measurement location, expressed in *cm*. The relief is the slope of the river trace in percent.

5.3.1.2 Discrete Covariables

We also have two accessory variables which are not continuous. The variable "grain" gives the size of the sediment grains, encoded as an ordered factor with three levels (fine, medium, and coarse grains, the typical sizes in which a sand is divided). Finally, "position" is a dichotomous variable describing whether the river belongs to the northern or southern part of the drainage basin. Both areas have slightly different geological characteristics, which influence the authors wanted to monitor.

```
> X4  = factor(Covariables$position)
> X3  = factor(Covariables$grain,c("f","m","c"),ordered=TRUE)
> class(X3)
```

```
[1] "ordered" "factor"
```

```
> X3
```

```
 [1] c c c m m m f f f c c c m m m f f f c c c m m m f f f c c c
[31] m m m f f f c c c m m m f f f c c c m m m f f f c c c m m m
[61] f f f c c c m m m f f f
Levels: f < m < c
```

We suspect that we will have different expected compositions for different levels of $X3$ or $X4$. We thus find ourselves in a similar situation to the ANOVA case (Sect. 5.1.2), but now the response is a composition.

X3 denotes an "ordered" covariable, coded as an ordered "factor" in **R**. Recall that the command levels provides the possible values the variable can take:

```
> levels(X3)

[1] "f" "m" "c"

> levels(X3) <- c("fine","medium","coarse")
> X3

 [1] coarse coarse coarse medium medium medium fine   fine
(...)
[65] coarse coarse medium medium medium fine   fine   fine
Levels: fine < medium < coarse
```

As in the real case, ANOVA is not specified in a unique way, and we have to choose the contrasts we want to use for estimation purposes. Given that $X3$ is ordered, the default contrast is a polynomial one, but we want to use instead the treatment contrast

```
> contrasts(X3)

              .L      .Q
[1,] -7.071e-01  0.4082
[2,]  4.351e-18 -0.8165
[3,]  7.071e-01  0.4082

> contrasts(X3)<-"contr.treatment"
> contrasts(X3)

       medium coarse
fine        0      0
medium      1      0
coarse      0      1
```

for reasons that will become clear in Sect. 5.3.3.

5.3.1.3 Multiple and Multivariate Covariables

Thus, the dataset contains a mixture of continuous and categorical covariables. We can thus study the dependence of composition on one or two continuous variables with regression, or else we can study the influence of the factors with ANOVA or finally get a complete model with both continuous and discrete covariables integrated.

In the next section, displaying the dependence, we will see that two different covariables (*relief* and *grain*) have a significant influence. This will give rise to several questions:

- *How do we model the joint influence of all relevant covariables?*
  What composition would be predicted if both relief and grain are known?
- *Is the joint influence an accumulation of independent linear effects, or do we have some interaction between the explanatory variables?*
  For instance, we may devise a regression model of composition on relief where the intercept is depending on grain size or where grain size modifies both slope and intercept.
- *Is one of the influences indirect?*
  For instance, maybe only the grain size has a direct influence to the composition, and the dependence on relief is inherited from an influence of this covariable on grain size.
- *Which of these many possible models is the best one?*
  A criterion might be that the model should be as complicated as necessary yet being as simple as possible (called the *principle of parsimony*, or *Occam's razor*). Another criterion might be that a model should predict *future* outcomes as precise as possible.

Using a complete linear modeling approach, we can answer all these issues in quite a straightforward fashion (Sects. 5.3.2–5.3.9). Choosing the best model among them will need a little bit more of discussion (Sect. 5.5).

5.3.2 Direct Visualization of the Dependence

To visualize the dependence of a composition on continuous covariables, we can use parallel plots, i.e., matrices of plots where the several explanatory variables occupy the same column, and the several representations of the explained composition use the same row. To display the role of a discrete covariable, we have two options: we either use the same parallel plots or use colors and symbols to distinguish the several levels.

5.3.2.1 Continuous Covariables

As happened with compositions as explanatory variables (Sect. 5.2.2), to visualize the dependence of a composition on one or more continuous covariables, we will need one dimension for the independent variable (to be plotted in the X-axis) and thus need to reduce the composition to one-dimensional projections. A starting point might be a clr, alr, or pairwise log-ratio transformation, but it is also possible to tailor a specific ilr basis from well-chosen balances (Sects. 2.5.5–2.5.7 and 4.3.1). The following code

```
> opar <- par(mar=c(4,4,1,1))
> pairwisePlot(cbind(X1,X2,logX1=log(X1)),clr(Y))
> par(opar)
```

generates a diagram quite similar to Fig. 5.9, but black-and-white and simpler. To obtain a plot of each possible pairwise log ratio against the explanatory variable, we could use the code in page 105, which would generate a matrix plot like Fig. 5.2, but with the log-ratio composition on the vertical axis.

5.3.2.2 Categorical Covariables Using Colors and Symbols

We can use symbols to represent the possible values of discrete covariable in some suitable compositional graphic, such as ternary diagrams, scatterplots of projections, and dot plots of projections (Fig. 5.7).

```
> opar = par(xpd=NA,no.readonly=TRUE)
> plot(Y,pch=c(3,20)[X4],col=c("red","green","blue")[X3])
> legend(x=0.5,y=0.45,abbreviate(levels(X4), minlength=1),
+    pch=c(3,20))
> legend(x=0.75,y=0.0,abbreviate(levels(X3), minlength=1),
+    pch=20,col=c("red","green","blue"),yjust=0)
> par(opar)
```

Symbols and color plots are meaningless without a legend. One can place the legend at some fixed coordinates (giving `x` and `y` of the upper left corner of the legend box) or else use the `locator` command to interactively locate the legend with the mouse:

```
> legend(locator(1),levels(X3),
        pch=20,col=c("red","green","blue"),xpd=NA,yjust=0)
```

Here `c(3,20)` encodes the plotting symbols to be used for each of the factors. Use the command `plot(1:40,1:40,pch=1:40)` to find the numbers of the plotting symbols. The `c("red","green","blue")` lists the colors to be used for each of the plotting levels of $X3$. See `? colours` for details. A shortcut with less control over the actual symbols and colors would be

```
> plot(Y,pch=as.numeric(X4),col=X3)
```

5.3.2.3 Categorical Covariables Through Parallel Plots

As with continuous covariables, we can also use the principle of parallel plotting to represent any one-dimensional projection of a composition, side by side in a panel for the various levels of the discrete covariable. We can use the set of all pairwise log

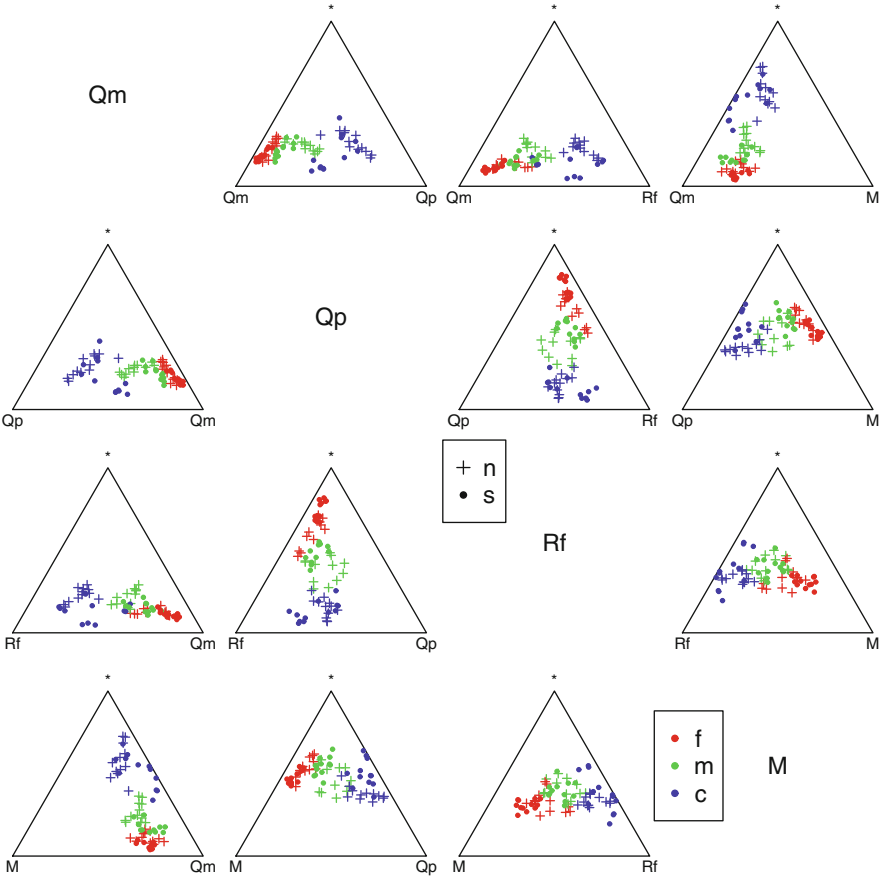


Fig. 5.7 Visualizing the dependence of a composition on discrete covariables with the help of *colors* and *symbols*

ratios or any standard log-ratio transformation (clr, alr, ad hoc ilr). To get a pairwise log-ratio representation, the compositional `boxplot` is the easiest tool (Fig. 5.8):

```
> boxplot(Y,X3,notch=TRUE)
```

The compositional boxplot as described in Sect. 4.1.3 is modified in two ways here. First, for each of the log-ratio, several boxplots are given, one for each level of the factor. This allows a qualitative and direct comparison of the distribution of all log ratios. Second, boxplots are now *notched* thanks to the `notch=TRUE` optional argument. These notches are constructed in a way that non-overlapping notches within the same panel can be interpreted as a strong hint that the true medians of the populations are different (Chambers et al., 1983). See the help `? boxplot.acomp`

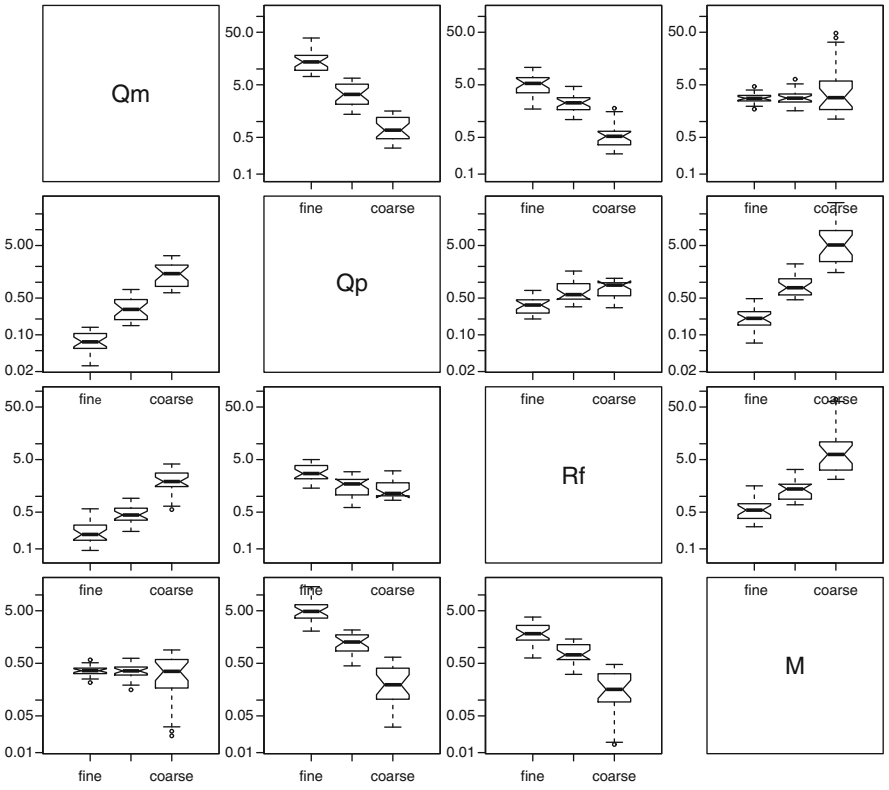


Fig. 5.8 A compositional boxplot splitted by factors

of the compositional boxplot for possible modifications, such as removing boxes, avoiding the logarithm, or displaying two-part subcompositions rather than (log) ratios.

5.3.2.4 Multiple and Multivariate Covariables

When several covariables of mixed nature are suspected to have an influence on the composition, we should try to visualize as much information as possible. We can then combine the principles of parallel plotting (for continuous covariables) and symbol/color (for discrete covariables):

```
> opar <- par(mar=c(4,4,0,0))
> pairwisePlot(cbind(X1,X2,logX2=log(X2)),clr(Y),
          pch=c(3,20)[X4],col=c("red","green","blue")[X3])
> par(opar)
```

Figure 5.9 shows the result of this command.

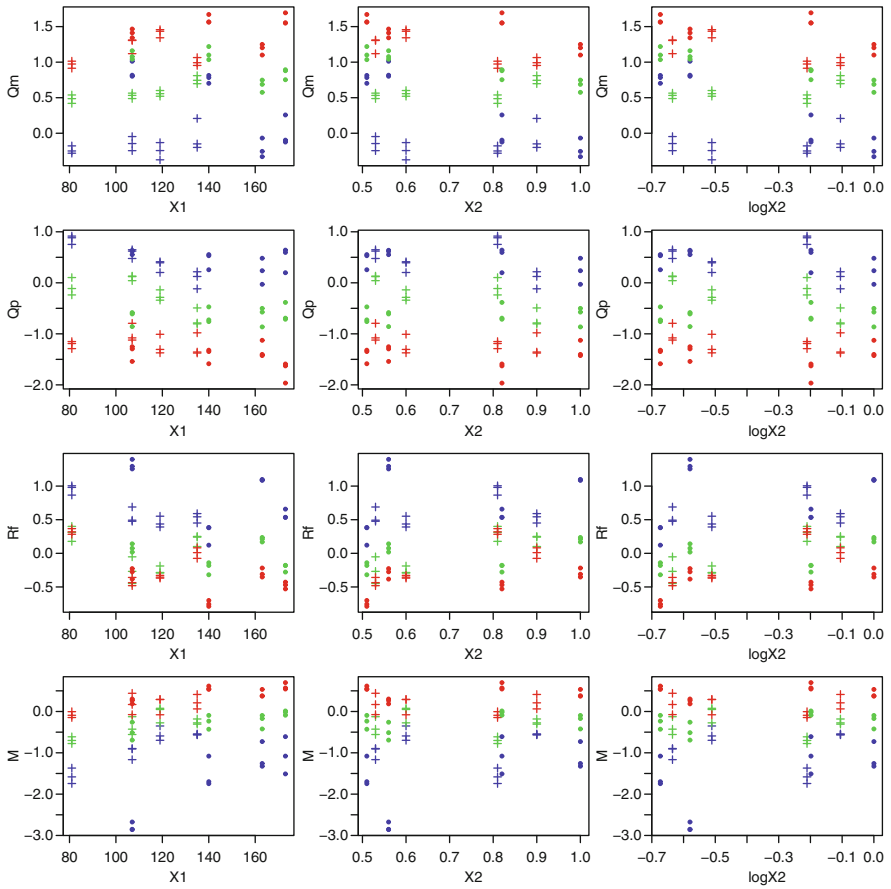


Fig. 5.9 A pairwise plot of the continuous covariables against the clr coefficients of the dataset. The two discrete covariables are added as *symbols* and *colors*, following the legend of Fig. 5.7

5.3.3 The Model and Its R Interface

5.3.3.1 The Regression Model

Based on the interpretation of multiple regression as a linear combination (5.2), we can build a regression model with compositional response and real independent variable using the linear Aitchison structure of the simplex:

$$\mathbf{Y}_i = \mathbf{a} \oplus X_i \odot \mathbf{b} + \varepsilon_i$$

where $\mathbf{a}$ and $\mathbf{b}$ are unknown compositional constants, $\mathbf{Y}_i$ is a random composition, X_i is a real random variable or design constant, and finally ε_i is a compositional random variable with null compositional expectation $\mathbb{1} = (1, \ldots, 1)/D$ and a

constant variance. Typically, we assume a ε to follow a normal distribution on the simplex, $\mathcal{N}_{\mathbb{S}}^{D}(\mathbb{1}, \Sigma)$, with clr-covariance matrix Σ.

To work with this model in **R**, we need to rewrite the model into a multivariate regression problem based on the principle of working in coordinates:

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{a} \oplus X_i \odot \mathbf{b} + \varepsilon_i \\ \text{ilr}(\mathbf{Y}_i) &= \text{ilr}(\mathbf{a}) + X_i \text{ilr}(\mathbf{b}) + \underbrace{\text{ilr}(\varepsilon_i)}_{\varepsilon_i} \end{aligned}$$

with a $\varepsilon_i \sim N(0_{D-1}, \Sigma_{\text{ilr}})$. This model is equal to (5.2) and can be estimated in **R** by the command

```
> (model = lm(ilr(Y)~log(X1) ))

Call:
lm(formula = ilr(Y) ~ log(X1))

Coefficients:
             [,1]    [,2]    [,3]
(Intercept)   2.543   1.108  -5.176
log(X1)      -0.696  -0.230   0.971
```

Note that `X1` was applied a log transformation (to honor its ratio scale). The output contains the estimated parameters $\hat{\mathbf{a}}$ (intercept) and $\hat{\mathbf{b}}$ (slope) in ilr coordinates. They can be displayed as compositions by

```
> ilrInv(coef(model),orig=Y)

             Qm       Qp     Rf     M
(Intercept) 0.01641 0.5984 0.3848 0.0003952
log(X1)     0.28218 0.1055 0.1302 0.4821039
attr(,"class")
[1] acomp
```

We would interpret the intercept $\mathbf{a}$ as the expected composition for $X = 0$. The slope $\mathbf{b}$ may be interpreted as the perturbation applied to the composition if X increases one unit. The output of the `summary` command for a truly multivariate model in ilr coordinates does not make much sense, as it tests each coordinate separately and not as a whole. However, we can use the `anova.mlm` to check for the joint significance of the influence of X:

```
> anova(model)

Analysis of Variance Table

            Df Pillai approx F num Df den Df Pr(>F)
(Intercept)  1  0.811     97.4      3     68 <2e-16
log(X1)      1  0.062      1.5      3     68   0.22
Residuals   70
```

In this example, the influence of X has a p-value larger than a typical α of, e.g., $\alpha = 0.05$. There is thus no evidence in the dataset that the discharge influences the composition of our sediments. On the contrary, if we try with relief,

```
> model=lm(ilr(Y)~log(X2))
> anova(model)

Analysis of Variance Table

            Df Pillai approx F num Df den Df  Pr(>F)
(Intercept)  1  0.813     98.5      3     68 < 2e-16
log(X2)      1  0.497     22.4      3     68 3.4e-10
Residuals   70
```

we find a significance below the critical level, and we can thus deduce that the composition notably depends on the relief of the basin.

5.3.3.2 The Compositional Analysis of Variance Model

Formally, the compositional analysis of variance model uses the same equation as in regression,

$$\mathbf{Y}_i = \mathbf{a} \oplus \mathbf{b}_{X_i} \oplus \varepsilon_i,$$

with $\mathbf{a}$ and $\mathbf{b}_g, g \in \{g_1, \ldots, g_m\}$ compositional constants in $\mathbb{S}_D$, and ε_i as usual a compositional random variable with neutral expected value *sorigin*, typically modeled as a normal distribution on the simplex $\mathcal{N}_{\mathbb{S}}^{D}(\mathbb{1}, \Sigma)$.

Recall that ANOVA models (Sect. 5.1.2) have an identifiability problem and that we must fix the contrasts we want to use for our factors. In general, we will prefer the treatment contrast. In that way, we will be able to interpret the intercept $\mathbf{a}$ as the expected response for the first level and the remaining constants $\mathbf{b}_{g_2}, \mathbf{b}_{g_3}, \ldots$ as the increments on average response from the first to the second level, from the first to the third, etc.

Thus, we have several compositional constants $a, b_{g_2}, b_{g_3} \ldots$, a different one for each group $g_i, i = 1, \ldots, m$. The different constants are distinguished by their lower index. Analogously to the regression model, we apply the ilr transformation to map the model to a multivariate linear model:

$$\text{ilr}\mathbf{Y}_i = \text{ilr}(\mathbf{a} \oplus \mathbf{b}_{X_i} \oplus \varepsilon_i) = \text{ilr}(\mathbf{a}) + \text{ilr}(\mathbf{b}_{X_i}) + \underbrace{\text{ilr}(\varepsilon_i)}_{\varepsilon_i}$$

with parameters:

```
> contrasts(X3) <- "contr.treatment"
> (model = lm(ilr(Y)~X3))

Call:
lm(formula = ilr(Y) ~ X3)
```

```
Coefficients:
             [,1]     [,2]     [,3]
(Intercept)  -1.8491  -0.2262   0.3383
X3medium      1.0044   0.0808  -0.6808
X3coarse      2.1000   0.5925  -1.8004
```

```
> (a = ilrInv(coef(model)[1,],orig=Y))
```

```
     Qm      Qp      Rf       M
0.60876 0.04454 0.12481 0.22189
attr(,"class")
[1] acomp
```

```
> (b = ilrInv(rbind(0,coef(model)[-1,]),orig=Y))
```

```
         Qm      Qp     Rf     M
         0.25000 0.2500 0.2500 0.2500
X3medium 0.11986 0.4961 0.2692 0.1148
X3coarse 0.03299 0.6429 0.3009 0.0232
attr(,"class")
[1] acomp
```

```
> (mu = a+b)
```

```
         Qm     Qp      Rf     M
         0.6088 0.04454 0.1248 0.2219
X3medium 0.4734 0.14335 0.2180 0.1653
X3coarse 0.2197 0.31323 0.4108 0.0563
attr(,"class")
[1] acomp
```

```
> (Sigma = ilrvar2clr(var(model)) )
```

```
         1        2        3        4
1  0.10751 -0.02454 -0.01613 -0.06684
2 -0.02454  0.07962  0.00914 -0.06422
3 -0.01613  0.00914  0.09206 -0.08507
4 -0.06684 -0.06422 -0.08507  0.21613
```

Here Σ is given in terms of a clr variance. We can then visualize the results as ellipses of the same shape and size centered around each of the expected values (see Sect. 5.3.4 for a more general case):

```
> plot(mu,pch=20)
> plot(Y,pch=".",add=TRUE)
> ellipses(mu,Sigma,2)
```

A test of the form

$$H_0 : \{b_{g_1} = \cdots = b_{g_m} = 0\} \text{ vs. } H_1 : \{\text{at least one } b_g \neq 0\}$$

is provided in the row linked to X3 of the ANOVA table:

```
> anova(model)

Analysis of Variance Table

            Df Pillai approx F num Df den Df Pr(>F)
(Intercept)  1  0.873    153.3      3     67 <2e-16
X3           2  1.002     22.7      6    136 <2e-16
Residuals   69
```

5.3.3.3 Linear Models with Several Main Effects

Multiple covariables can be added into a linear model simply by adding several of the influence terms into the modeling equation

$$\mathbf{Y}_i = \mathbf{a} \oplus \log(X1_i) \odot \mathbf{b}_1 + X2_i \odot \mathbf{b}_2 \oplus \mathbf{b}_3(X3) \oplus \mathbf{b}_4(X4) \oplus \varepsilon_i \tag{5.4}$$

for continuous covariables $X1$ (in this case, a ratio scale) and $X2$ (real scale) and discrete covariables $X3$ (ordinal) and $X4$ (categorical/dichotomous). If influence terms of different variables are added, we speak about *main effects*. Correspondingly, the model would be expressed in **R** by adding the terms into the formula of the model. The system recognizes the factors automatically from their class:

```
> (model = lm(ilr(Y)~log(X1)+X2+X3+X4 ))

Call:
lm(formula = ilr(Y) ~ log(X1) + X2 + X3 + X4)

Coefficients:
             [,1]     [,2]     [,3]
(Intercept) -0.1471   2.1508  -7.9290
log(X1)     -0.3903  -0.7144   1.8125
X2           0.4939   1.4462  -0.2496
X3medium     1.0044   0.0808  -0.6808
X3coarse     2.1000   0.5925  -1.8004
X4south     -0.3438   0.0711  -0.6047
```

The output now contains the estimated parameters in ilr coordinates: the intercept $\hat{\mathbf{a}}$, the slopes $\hat{\mathbf{b}}_1$, $\hat{\mathbf{b}}_2$ for each continuous covariable, and the ANOVA effects $\hat{\mathbf{b}}_3(X3)$ and $\hat{\mathbf{b}}_4(X4)$, with one row for each level beyond the first. The first level of the factor is assumed to have an effect of zero due to the implicit usage of the `contr.treatment` contrasts.

5.3.4 Estimation and Representation of the Model Parameters

5.3.4.1 Separate Representation of Each Coefficient

The estimated parameters of a compositional linear model can be extracted with the `coef` command

```
> coef(model)
```

```
               [,1]     [,2]    [,3]
(Intercept) -0.1471  2.15081 -7.9290
log(X1)     -0.3903 -0.71445  1.8125
X2           0.4939  1.44616 -0.2496
X3medium     1.0044  0.08075 -0.6808
X3coarse     2.1000  0.59254 -1.8004
X4south     -0.3438  0.07105 -0.6047
```

Here each line corresponds to the ilr transformation of the compositional parameter. The corresponding compositions can be reconstructed by

```
> (coefs <- ilrInv(coef(model),orig=Y))
```

```
               Qm       Qp       Rf       M
(Intercept) 0.06960 0.05653 0.87386 1.594e-05
log(X1)     0.15412 0.08874 0.04875 7.084e-01
X2          0.07540 0.15159 0.62839 1.446e-01
X3medium    0.11986 0.49610 0.26920 1.148e-01
X3coarse    0.03299 0.64291 0.30090 2.320e-02
X4south     0.34818 0.21412 0.29787 1.398e-01
attr(,"class")
[1] acomp
```

where the `orig` parameter is used to recover the variable names, which are lost in the ilr transformation. The parameters can be represented as compositions by (Fig. 5.10)

```
> barplot(coefs,las=2,xlim=c(0,11))
```

If we want to display the uncertainty of these parameters, we need their estimation variance. The estimated estimation variance is given by the `vcov` command

```
> vcov(model)
```

This ignores the matrix structure of the coefficients and treats them as a vector. However, we can extract the compositional variances of individual parameters by reshaping this variance matrix, with an ad hoc function `vcovAcomp`, and plot them as confidence regions for each of parameters into ternary diagrams (Fig. 5.11):

```
> vars <- vcovAcomp(model)
> dim(vars)
```

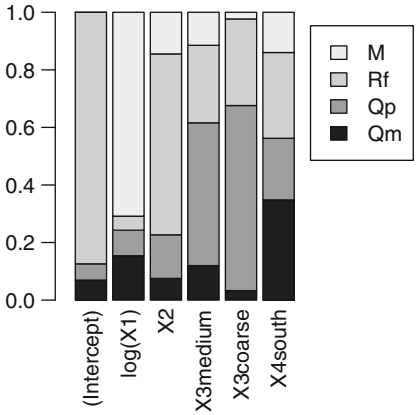


Fig. 5.10 A bar chart representation of the parameters of a linear model with compositional response and several main effects. Each bar is associated to a continuous covariable or level of a categorical covariable

```
[1] 3 3 6 6
```

```
> alpha=0.05
> plot(coefs,pch=as.character(1:nrow(coefs)))
> plot(coefs-coefs,pch=20,add=TRUE)
> for(i in 1:nrow(coefs)){
+   ellipses(acomp(coefs[i,,drop=FALSE]),
+            ilrvar2clr(vars[,,i,i]),
+            r=ConfRadius(model,1-alpha),
+            col="gray")
+   }
> par(xpd=FALSE)
```

The `vcovAcomp` command reorganizes the coefficient matrix in such a way that `vcovAcomp[ , ,i,j]` is the ilr covariance of the ith and the jth parameter. Thus, `vcovAcomp[ , ,i,i]` is the estimation variance $\widehat{\text{var}}_{\text{est}}(\hat{\mathbf{b}}_i)$ of the ith parameter expressed in ilr. The `ConfRadius` command allows to compute the radius of the confidence regions of the model parameters, such that under the assumptions of normality on the simplex for the residuals, the true value of the parameter $\mathbf{b}_i$ is within the ellipse around $\hat{\mathbf{b}}_i$ with a `prob` probability. This is based on the fact that the estimated Mahalanobis distance follows a Hotelling's T^2 distribution, with dimension $D-1$ and degrees of freedom given by the residuals of the linear model (Fahrmeir et al., 1996):

$$\text{ilr}(\hat{\mathbf{b}}_i \ominus \mathbf{b}_i) \cdot \widehat{var}_{\text{est}}^{-1}(\mathbf{b}_i) \cdot \text{ilr}^t(\hat{\mathbf{b}}_i \ominus \mathbf{b}_i) \sim T^2(D-1, \texttt{df.residual}).$$

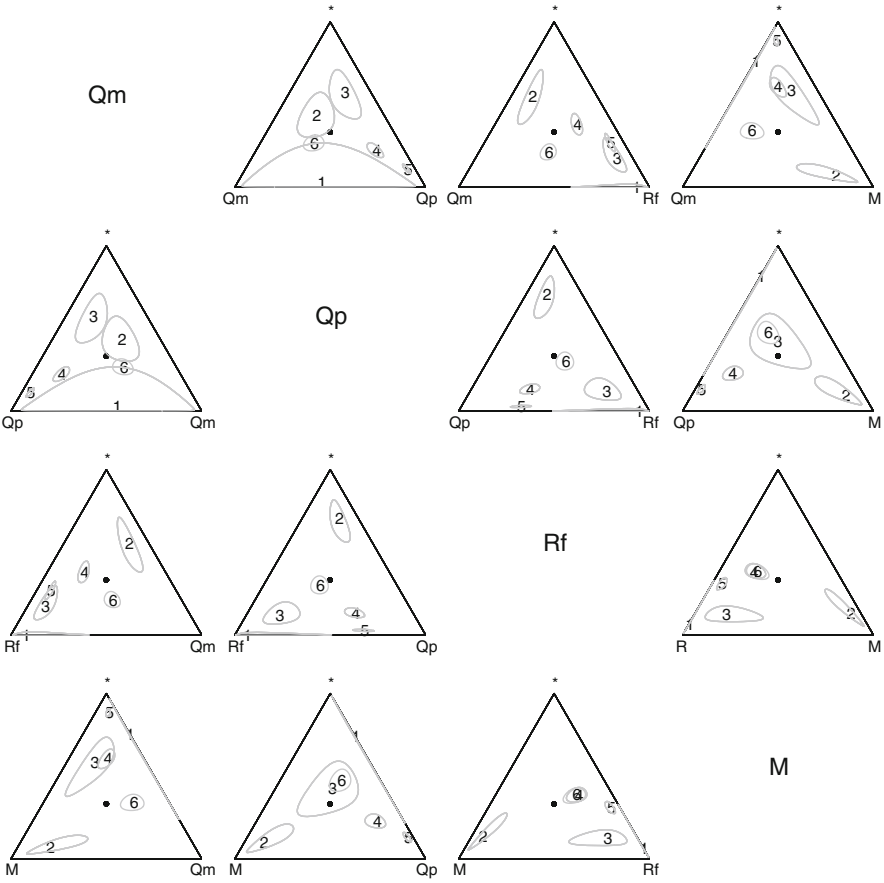


Fig. 5.11 Parameter estimates and 95 % confidence ellipsoids of a linear model with several main effects. Legend: 1 = (Intercept); 2 = log(X1); 3 = X2; 4 = X3medium; 5 = X3coarse; 6 = X4south. A variable may be removed from the model if the confidence ellipse around its parameter always contains the barycenter

5.3.4.2 Joint Representation of the Whole Model Coefficients

An alternative way to see a model with several main effects, like the one in (5.4), is as a linear mapping

$$\text{ilr}(\mathbf{Y}) = \text{ilr}(\mathbf{a}) + \mathbf{X} \cdot \mathbf{B} + \varepsilon_i$$

where now $\mathbf{B}$ is a square matrix space representing a linear transformation from the space of explanatory variables to the simplex (and expressed in ilr for the compositional side). $\mathbf{X}$ contains all the explanatory variables, and the rows of $\mathbf{B}$ are the ilr coordinates of the several vectors $\mathbf{b}_i$.

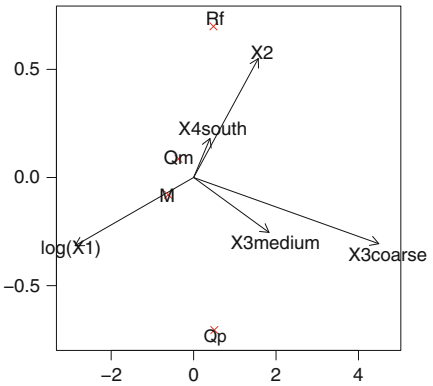


Fig. 5.12 Biplot representation of the linear model. Each *arrow* represents a one-unit change in the direction of each explanatory variable. Projecting these *arrows* onto the links between the compositional variables, we obtain the change in the response composition predicted by the linear model

With this interpretation, we can apply a singular value decomposition (SVD) to understand the structure of $\mathbf{B}$. A SVD decomposes a $P \times C$ matrix into three matrices:

$$\mathbf{B} = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{V}^t \tag{5.5}$$

where $\mathbf{U} = (\mathbf{u}_1 \cdots \mathbf{u}_R)$ and $\mathbf{V} = (\mathbf{u}_1 \cdots \mathbf{u}_R)$ are matrices with orthogonal columns $\mathbf{u}_i \in \mathbb{R}^P$ and $\mathbf{v}_i \in \mathbb{R}^C$ and $\mathbf{D} = (d_i \delta_{ij})_{ij} = \text{diag}(\mathbf{d})$ is a $R \times R$ diagonal matrix, with $R = \min(P, C)$. Note that in the present case, $P = 5$ the number of estimated vectors, and $C = D - 1 = 3$ the number of ilr coordinates of our composition (and in this case, $R = 3$). These elements are, respectively, called left and right singular vectors and singular values. If we multiply $\mathbf{B}$ with the ith left singular vector $\mathbf{u}_i$, we get

$$\mathbf{u}_i \cdot \mathbf{B} = \mathbf{u}_i \cdot \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{V}^t = \mathbf{e}_i \cdot \mathbf{D} \cdot \mathbf{V}^t = (d_i \mathbf{e}_i) \cdot \mathbf{V}^t = d_i (\mathbf{e}_i \cdot \mathbf{V}^t) = d_i \mathbf{v}_i^t.$$

Thus, if we change the explanatory variables one unit along the direction of $\mathbf{u}_i$, then we move the explained composition a distance of d_i along the direction of $\mathbf{v}_i$.

The SVD is also the underlying technique behind the biplot (Sects. 6.1.1 and 6.1.2). We can thus take profit of this decomposition to graphically represent the whole linear mapping obtained with regression (Fig. 5.12). If the sum of the first two squared singular values is large enough (compared with the sum of all of them), then the following biplot will be a good representation of the global regression model:

```
> B = clr(coefs[-1,])
>  svdlm = svd(B)
> opar <- par(xpd=NA,mar=c(2,2,0,0))
> coloredBiplot(svdlm, scale =0, xarrows=TRUE, ypch=4,
                  ynames=colnames(B), xnames=rownames(B))
> sum(svdlm$d[1:2]^2)/sum(svdlm$d^2)

[1] 0.9095

> par(opar)
```

Note that this biplot summarizes a very high proportion of the variability described by the model; thus we can trust the following interpretations.

In general terms, an arrow represents a one-unit change in the direction of an explanatory variable: for instance, the arrow $X2$ represents a change from a relief of 1 to a relief of 2, and the arrow $X3$*medium* the change from fine sand to medium sand. On the other hand, a link between two compositional variables (e.g., the segment joining M and Qm) represents a one-unit change in the log ratio of the two parts: in this case, adding $+1$ to $(1/\sqrt{2})\ln(Qm/M)$, for instance, passing from a ratio Qm/M of 1:1 to a ratio of $\approx$4:1. Projecting the arrow of an explanatory variable onto the direction of each of the compositional links, we find (an approximation of) the expected change in the composition caused by a one-unit increment of the explanatory variable. Thus, passing from basin north to basin south (a one-unit change along $X4$*south*) increases the ratio Qm/M by a factor of 4. Another example is: if we project $X3$*medium* onto the link $M - Qp$, the resulting arrow is approximately one half of the link; thus, passing from fine to medium sand, we should expect the ratio Qp/M to increase by a factor of two:

$$+\frac{1}{2} = \frac{1}{\sqrt{2}} \ln \frac{Qp}{M} \quad \Rightarrow \quad \frac{Qp}{M} = \exp\left(+\frac{\sqrt{2}}{2}\right) \simeq 2.$$

5.3.5 Testing the Influence of Each Variable

The `anova` command is used to test that the role of each variable is significant:

```
> anova(model)

Analysis of Variance Table

            Df Pillai approx F num Df den Df  Pr(>F)
(Intercept)  1  0.932    292.1      3     64 < 2e-16
log(X1)      1  0.511     22.3      3     64 5.2e-10
X2           1  0.723     55.7      3     64 < 2e-16
X3           2  1.151     29.4      6    130 < 2e-16
X4           1  0.382     13.2      3     64 8.4e-07
Residuals   66
```

However, when several covariables are involved, we must keep in mind that each line of the table corresponds to a test of the form:

- H_0, null hypothesis: given the preceding $(i-1)$ variables, the ith covariable has no influence. If this is a continuous covariable, we can say that the slope coefficient $\mathbf{b}_i = \mathbb{1}$. If this is a factor, we can say that the composition has the same expectation for all levels.

- H_1, alternative hypothesis: the independent variable of the line is needed, additionally to all preceding variables, i.e., its coefficients $\mathbf{b}_i \neq \mathbb{1}$.

The p-value of the test can be found in the last column of the table. According to the principle that the model should be as simple as possible, a variable should be removed if its influence is not significant. To understand why a nonsignificant parameter should be removed, we can consider the following: for a nonsignificant test, we did not prove the parameter to be different from zero. In particular we do not know in which direction it influences the compositions. Used in prediction, that might well increase the prediction error. Therefore, it is safer to remove it.

A variable is only needed in the model if it is necessary given that the information from all the other covariables is already used. It is thus necessary to check each covariable as last term in the model.

Currently, there is no way to automatize this, since the built-in **R** command `drop1` is not functional for the `"mlm"` model. We need to manually enumerate each covariable in the last position and extract the test of the fifth row of the ANOVA table:

```
> anova(lm(ilr(Y)~X2 + X3 + X4 + log(X1))) [5,]

Analysis of Variance Table

        Df Pillai approx F num Df den Df  Pr(>F)
log(X1)  1  0.563     27.5      3     64 1.5e-11

> anova(lm(ilr(Y) ~ log(X1) + X3 + X4 + X2)) [5,]

Analysis of Variance Table

   Df Pillai approx F num Df den Df Pr(>F)
X2  1  0.704     50.7      3     64 <2e-16

> anova(lm(ilr(Y) ~ log(X1) + X2 + X4 + X3)) [5,]

Analysis of Variance Table

   Df Pillai approx F num Df den Df Pr(>F)
X3  2   1.15     29.4      6    130 <2e-16

> anova(lm(ilr(Y) ~ log(X1) + X2 + X3 + X4)) [5,]

Analysis of Variance Table

   Df Pillai approx F num Df den Df  Pr(>F)
X4  1  0.382     13.2      3     64 8.4e-07
```

In this case, all covariables are significant at a 5 % level, when added last. A different approach to covariable selection and elimination will be discussed in the section about model selection (Sect. 5.5.3).

5.3.6 Comparing Predicted and Observed Values

When the parameters are estimated, we can try to find a "noise-free" version of $\mathbf{Y}$, by evaluating the model equation without the error term plugging the given values for the independent variables and the estimated parameters as coefficients. For instance, in a model with `log(X1)` and `X3` as covariables, we would have

$$\hat{\mathbf{Y}}_i = \mathbf{a} \oplus \log(X1_i)\hat{\mathbf{b}}_1 \oplus \hat{\mathbf{b}}_{X3}$$

$\hat{\mathbf{Y}}_i$ is called the *predicted value*. The predicted values can be computed by the `predict` command and need to be ilr-back-transformed into compositions:

```
> Pred = ilrInv(predict(model),orig=Y)
```

The predicted values are again compositions and may be visualized together with the original data, either as families of points or with segments joining predictions and observations:

```
> plot(Y,pch=".")
> plot(Predicted,add=TRUE,pch=20)
> segments(x0=Predicted,y=Y)
```

Note that these plots can be very noisy if the dataset is of medium size. A better representation may be obtained plotting the actually observed values against the predicted values in a family of one-dimensional projections. We could use any coordinate log-ratio transformation (alr,clr,ilr) or else the set of all pairwise log ratios (Fig. 5.13):

```
> opar <- par(mfrow=c(4,4), mar=c(2,2,1,1), oma=c(4,4,0,0))
> Ypred = ilrInv(predict(model),orig=Y)
> for(i in 1:4){
+  for(j in 1:4){
+   plot(log(Y[,i]/Y[,j]),
+        log(Ypred[,i]/Ypred[,j]),
+        pch=ifelse(i!=j,1,""))
+   if(i==j){text(x=0,y=0,
+           labels=colnames(Y)[i],cex=1.5)
+   }else{
```

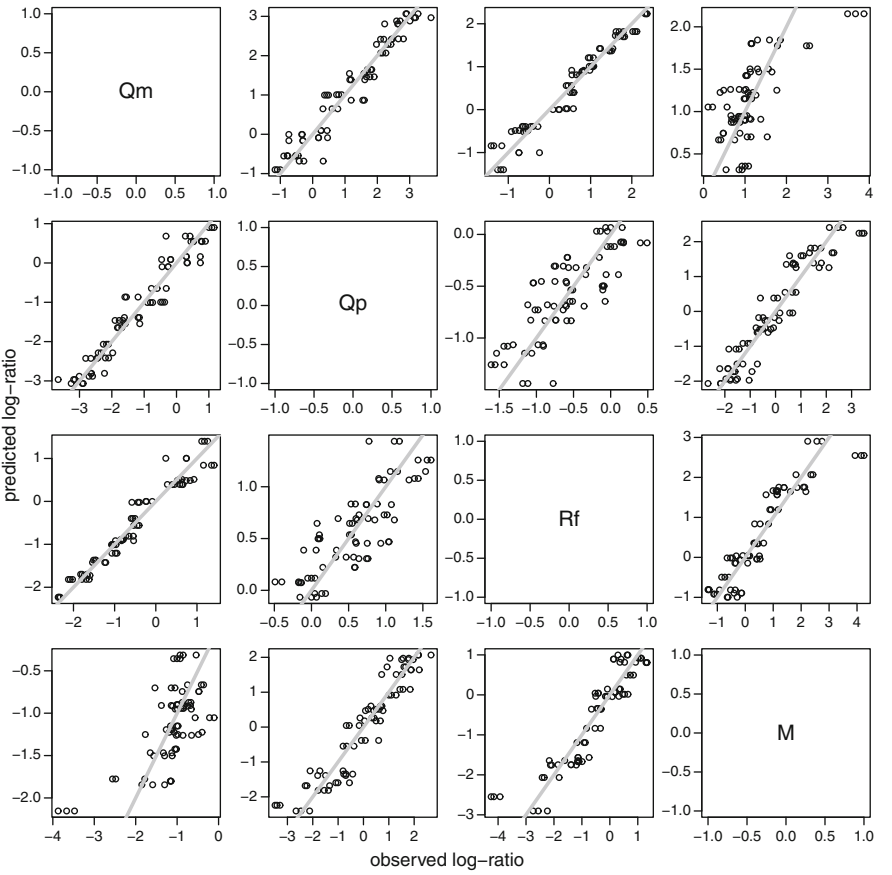


Fig. 5.13 Observed against predicted values of the composition, as obtained with the estimated model. Each diagram represents the pairwise log ratios of observed against predicted values, of the variable in the row divided by the variable in the column

```
+      abline(a=0,b=1,col="gray",lwd=3)
+    }
+  }
+ }
> mtext(text=c("observed log-ratio","predicted log-ratio"),
                    side=c(1,2),at=0.5,line=2,outer=TRUE)
> par(opar)
```

Again the trick of giving `pch=ifelse(i!=j,1,"")` allows us to comfortably get empty diagonal panels. In this plot we expect to see a clear concentration of dots around the reference line, the identity. The main use of these plots is to visualize the dependence and its strength. In our case, all plots show a more or less simple linear

dependence. Only those ratios involving low values of M represent an exception, because they are badly predicted by the model.

5.3.7 Prediction of New Values

To predict unobserved values of the dependent variable from observations of the independent variables, we also use the `predict` command in **R**, but giving as `newdata` argument the new values of the covariables. This function evaluates the model equation with the given values for the independent variables and the estimated parameters as coefficients and without error term. The term *prediction* denotes the results of `predict` for the given `newdata` values, to distinguish them from the predicted values (the results obtained with `predict` when no `newdata` is given). The new values for the independent variables can be provided in the argument `newdata` as a list or if they do not contain multivariate objects as a `data.frame`.

```
> newdata=list(X1=geometricmean(X1),
+   X2=mean(X2),
+   X3=factor("coarse"),
+   X4=factor("north"))
```

This would predict the expected composition (in ilr) for coarse sediments from a hypothetical basin placed in the northern part of the study area, which discharge would be equal to the geometric average of the data and its relief would be the average relief. Since for **R** the model is stated in ilr coordinates, we need to back-transform the predictions into compositions with the `ilrInv` function

```
> prediction <- ilrInv(predict(model,newdata=newdata),orig=Y)
```

There are three types of errors associated to a prediction:

- *Estimation error*
  The random difference to the optimal prediction due to the estimation errors in the estimated coefficients.
- *Prediction error*
  The random difference to the true unobserved values due to estimation errors in the estimated coefficients and due to the variability of the true value around its conditionally expected value.
- *Model error*
  Our models, the good and the bad ones, cannot be as complex as the reality they try to "model." This systematic inability of the model to fit the unknown reality induces an equally systematic difference of prediction with optimal parameters to the true conditional expectation.

The first two errors are typically quantified by variances (estimation variance, prediction variance), squared error (mean-squared estimation error or mean-squared prediction error), and regions (confidence intervals/regions, predictive regions).

Though both fundamental from philosophical and practical points of view, the model error is (strangely) mostly ignored in statistical practice.

For predictions, we distinguish two types of regions. The predicted value itself is uncertain only due to the fact that regression parameters are not known but estimated. The region built in this case is typically called a *confidence region*. If we consider the regression parameters as fixed, then we can also build what is called a *prediction region*, which uncertainty comes from the variance Σ_ε of the residuals of the model. In the case of a one-dimensional response, confidence and prediction intervals are provided by the standard **R** command `predict` (page 98). For multivariate response, e.g., compositions of more than one ilr coordinate, that function is useless, because the actual uncertainty regions are ellipses.

Moreover, we should accumulate *both* uncertainty sources, one coming from parameter estimation and the other coming from the residual variance. This provides the *predictive regions*. These two variance sources are, respectively, provided (in the ilr coordinates used) by the `vcov` command and the `var.mlm` method of the `var` command.

The residual variance is available in "`compositions`" in a form directly usable for our computations:

```
> (varEpsilon = var(model))
```

```
         [,1]      [,2]      [,3]
[1,]  0.06101 -0.010549  0.005610
[2,] -0.01055  0.030774 -0.003159
[3,]  0.00561 -0.003159  0.184500
```

On the contrary, we saw in Sect. 5.3.4 that the parameter estimation variance matrix is provided in quite a complex form. We must first then "propagate" that variance on the regression parameters as uncertainty on the predicted values themselves. For this goal, it is convenient to take all variables, variances, and parameters expressed in ilr coordinates, which transforms the regression equation in a matrix-multiplication form like (5.1). The prediction is thus an affine linear mapping of the matrix **B**. Then to propagate its variance, it is sufficient to apply the same linear mapping on both sides of this variance. This is done by the following code:

```
> getModelMatrix <- function(object,
                      newdata=NULL,na.action=na.pass) {
+   if( is.null(newdata) )
+     return(model.matrix(object))
+   Terms <- delete.response(terms(object))
+   mf <- model.frame(Terms,newdata,na.action=na.action,
                                  xlev = object$xlevels)
+   if (!is.null(cl <- attr(Terms, "dataClasses")))
+     .checkMFClasses(cl, mf)
+   model.matrix(Terms, mf, contrasts.arg = object$contrasts)
+ }
> (X <- getModelMatrix(model,newdata))
```

```
  (Intercept) log(X1)     X2 X3medium X3coarse X4south
1           1   4.827 0.7163        0        1       0
attr(,"assign")
[1] 0 1 2 3 3 4
attr(,"contrasts")
attr(,"contrasts")$X3
[1] "contr.treatment"

attr(,"contrasts")$X4
[1] "contr.treatment"
```

```
> XX <- kronecker(diag(ncol(predict(model))),X)
> (estVar = XX %*% vcov(model) %*% t(XX))
```

```
            [,1]       [,2]       [,3]
[1,]  0.0039348 -0.0006803  0.0003618
[2,] -0.0006803  0.0019847 -0.0002037
[3,]  0.0003618 -0.0002037  0.0118991
```

```
> (predVar = estVar+varEpsilon)
```

```
          [,1]      [,2]      [,3]
[1,]  0.064946 -0.011229  0.005972
[2,] -0.011229  0.032759 -0.003363
[3,]  0.005972 -0.003363  0.196399
```

The first lines define a function that organizes the new values of the explanatory variables in a matrix with the adequate dimensions, the so-called model matrix. Then we apply the model matrix by both left and right multiplication to the estimation covariance matrix. The result is the sought estimation variance of the predicted values. Finally, we add both uncertainty components together. It is now possible to draw the predictions together with their predictive regions into ternary diagrams (Fig. 5.14):

```
> alpha=0.05
> plot(Y,pch=".")
> plot(prediction,pch=20,add=TRUE)
> ellipses(prediction,ilrvar2clr(estVar),
+     r=ConfRadius(model,1-alpha))
> ellipses(prediction,ilrvar2clr(predVar),
+     r=ConfRadius(model,1-alpha))
```

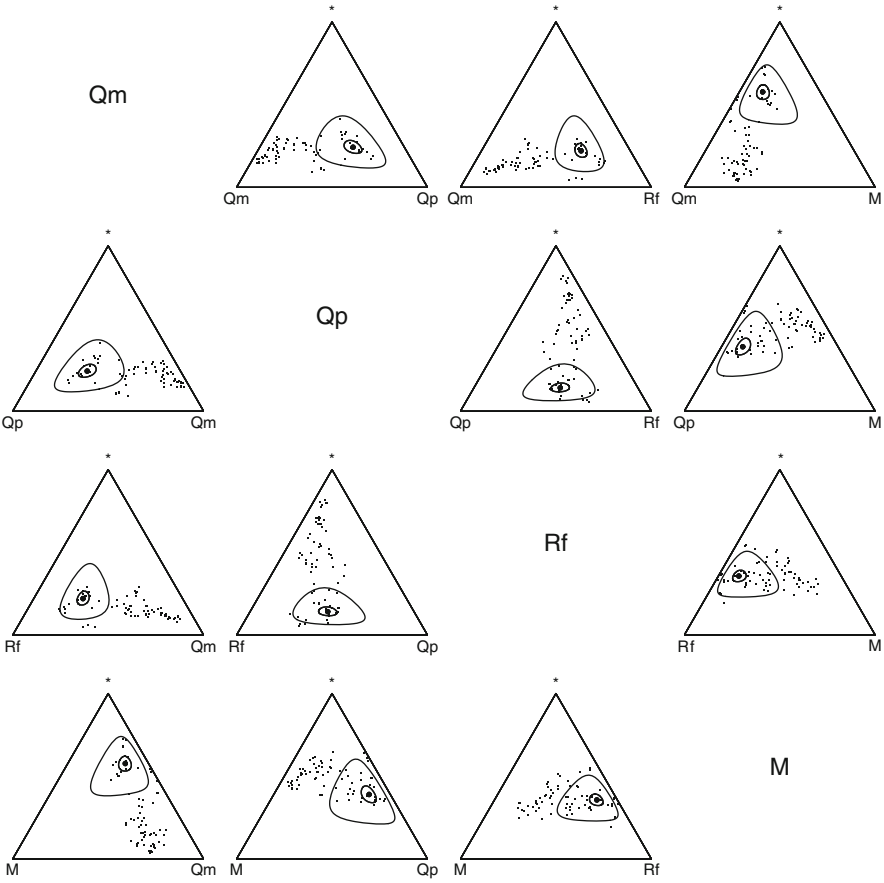


Fig. 5.14 Ternary diagrams showing the confidence and predictive regions around the predicted composition, marked with a *circle*, for some fixed values of the explanatory variable (coarse sand, northern subbasin, average discharge, and relief). The smaller ellipsoid corresponds to the estimation error variance of the predicted values. The larger ellipse corresponds to the predictive area, in which a new prediction with this combination of covariable values would fall with a probability of $1 - \alpha = 0.95$. The *dots* show the dataset, for reference purposes

5.3.8 Residuals

5.3.8.1 Definition and Computation

Let us go back to the predicted values, where the dependent variable Y_i has actually been observed. The predicted values are the compositions that the model would suggest as expected for the given set of parameters. The difference between these predicted values $\hat{\mathbf{Y}}_i$ and the values actually observed $\mathbf{Y}_i$ are called `residuals`

$$\hat{\varepsilon}_i = \mathbf{Y}_i \ominus \hat{\mathbf{Y}}_i = \text{ilr}^{-1}(\text{ilr}(\mathbf{Y}_i) - \widehat{\text{ilr}(\mathbf{Y}_i)})$$

and can thus be obtained with

```
> Resid = ilrInv(resid(model),orig=Y)
```

The residuals are not exactly equal to the errors ε_i of the model, but their best approximation we have available. We should then check the modeling hypotheses about the regression error on these estimated residuals. Recall that we assumed them to be approximately normally distributed and mutually independent and show a constant variance (homoscedastic). The next subsections cover a descriptive analysis of the residuals to visually check these issues. Note that it is also possible to apply the methods for outlier classification provided in Sect. 7.4, though we will not repeat that here.

5.3.8.2 Visual Inspection of Residuals

Since residuals are also compositions, they can again be plotted in ternary diagrams (Fig. 5.15):

```
> plot(Resid, cex=0.5)
> #title("Ternary Diagrams of Residuals",outer=TRUE,line=-1)
```

This representation is particularly suited to visually compare the residual spread of different models or along different directions, because residuals are always centered and because ternary diagrams keep their scale. In the example of Fig. 5.15, we see fairly comparable spreads, between any pair of components or along the balance in the upper corner of each ternary diagram (the geometric mean of the remaining parts). The ternary diagram of the residuals can also reveal some structures in the residuals, e.g., when an important categorical covariable had not been considered. In the example, we can observe a clear group of three observations with a very small amount proportion of M that was also singularized in the pairwise log ratios of predicted against observed values (Fig. 5.13). If the variance of the residuals is small, one should consider to plot a scaled version of the residuals:

```
> plot(4*Resid)
```

5.3.8.3 Checking Symmetry and Normality

The test theory behind linear models is based on the assumption that residuals have a normal distribution on the simplex (Sect. 3.1.1). As explained there, a vectorial variable is normally distributed if all its univariate projections are normally distributed. We cannot check all univariate projections, but the set of all pairwise log ratios provides a canonical set of $D(D-1)/2$ directions, which allows to get a good global impression. The assumption of normality can thus be visually checked

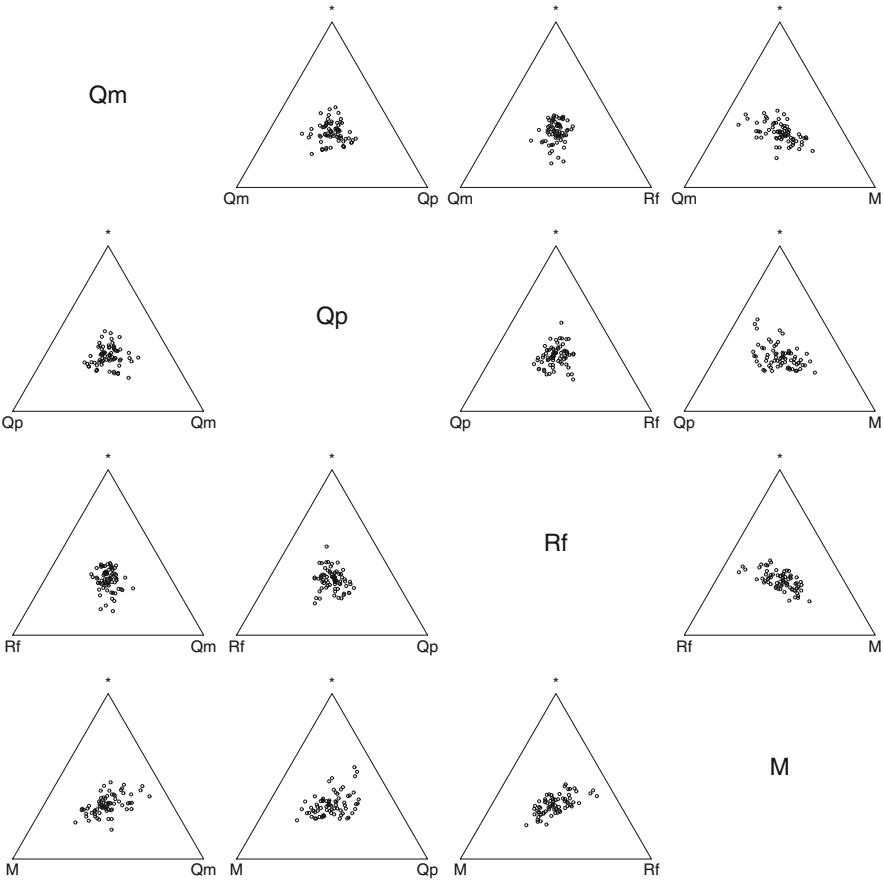


Fig. 5.15 Ternary diagrams of residuals of the linear model. Recall that residuals are always centered. It is thus easy to visually compare the variability among different models and different subcompositions

on each of these possible log ratios, for instance, with boxplots for compositions (Fig. 5.16):

```
> boxplot(Resid)
```

Recall that this command was presented in Sect. 4.1.3. The strength of boxplots is to reveal asymmetry (when the box itself is not symmetric) and the existence of outliers (since they are plotted as dots quite beyond the whiskers end). In the example (Fig. 5.16), we again see several outliers in ratios involving a low value of the M component.

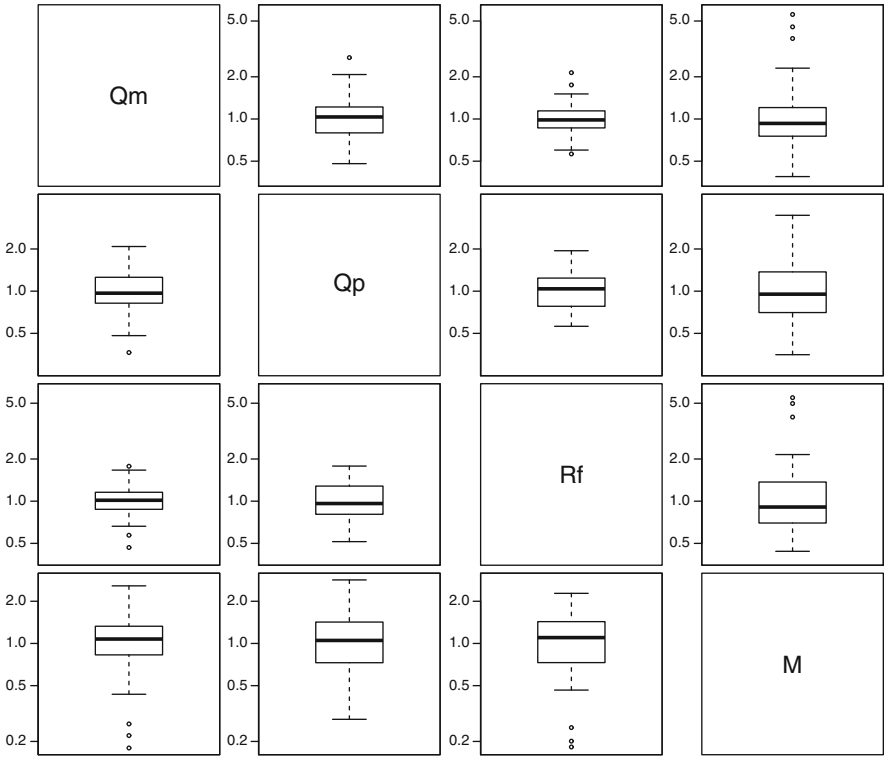


Fig. 5.16 Boxplot of the residuals of the linear model. Each frame shows a boxplot of the log ratio of the component in the row against the component in the column. This diagram helps in assessing symmetry of the residuals and detecting outliers. In this application, the outliers in the components involving M show up clearly

To compare the shape of the distribution to the theoretical normal distribution on the simplex, we could use a Quantile–Quantile plot, applying the `qqnorm` command (Fig. 5.17):

```
> qqnorm(Resid)
```

In a QQ plot, a dataset is considered to follow a normal distribution if it plots in a straight line against the quantiles of a standard normal distribution. The intercept of the line is the mean and the value at abscissa one would correspond to the standard deviation. In this case, we need to compare several lines, for each log ratio. Again the diagrams involving component M show a clear deviation from a straight line, though in this case they involve some more points apart of those already identified as outliers. This might be a hint towards a heavy-tailed distribution.

Both figures show some deviations from normality. It would be thus interesting to proceed to an actual numeric test of normality, as is treated in Chap. 3. Accepting or rejecting these tests might have several consequences for the analysis.

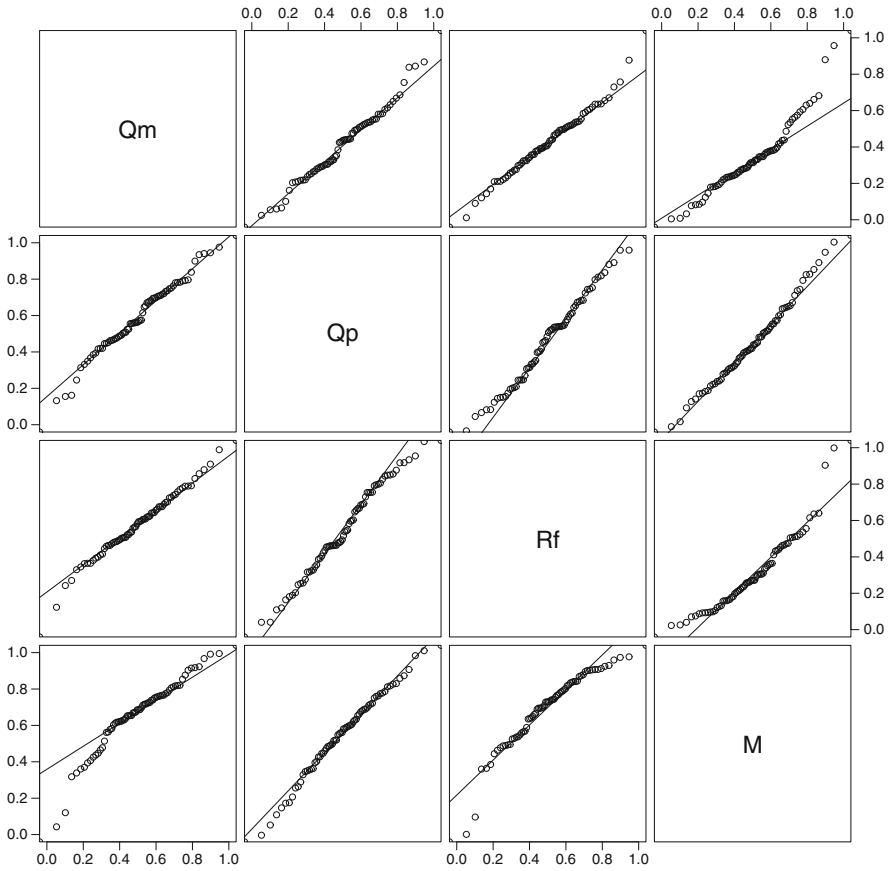


Fig. 5.17 Quantile–Quantile plots comparing the distribution of residuals against a normal distribution on the simplex, for each possible pairwise log ratio. The lines are robustly fitted references. Note that only the log ratio of *Qm* and *M* shows a clear deviation from normality, in the direction of a heavy-tailed distribution

- If all deviations come from the presence of outliers and they can be considered as wrong observations, we should reduce their influence on the parameter estimates, as outliers usually have large levering effects on the fitting of linear models. A robust fitting would be necessary. Unfortunately, a robust fitting of multivariate linear models is not available in **R** at the time of writing of this book.
- If the outliers are considered as genuine values that should be included in the linear model, we may be confident that our parameter estimates and further predictions are still unbiased, as long as we assume the existence and relevance of compositional means and metric variances, i.e., correct as a least-squares solution (Daunis-i Estadella et al., 2002).
- Note however that most inferential procedures, such as tests and confidence intervals, are based on normality assumptions. In the context of simple univariate

linear models, we could now try rank-based tests. Unfortunately, ranks are meaningless in multivariate data due to the lack of a natural order. We may thus need to understand the consequences of assuming non-normal distributions for the residuals. This is discussed in Sect. 5.5.4.

5.3.8.4 Checking Homoscedasticity

The theory of linear models is further based on the assumption that the residuals ε_i have constant variance for any combination of values of the explanatory variables. The estimation through unweighted least squares is optimal if the variances are equal. Tests can get more aggressive (i.e., reject with a higher probability than α) if the variance varies substantially between different values of the independent variables. In all exploratory comparison of variability, we should however keep in mind that variances behave quite like a ratio scale (i.e., only orders of magnitude matter) that the human eye cannot easily judge.

A standard tool in linear models is to plot residuals against predicted values in a scatterplot. This reduces the multiple covariables to one optimal regressor. In the compositional case, we have a multidimensional prediction and multidimensional residuals. We must thus first project them to a family of log ratios, for instance, the clr (Fig. 5.18)

```
> opar <- par(oma=c(3,3,0,0),mar=c(4,4,1,0))
> pairwisePlot(clr(Pred),clr(Resid))
> mtext(text=c("predicted values (clr)","residuals (clr)"),
                    side=c(1,2),at=0.5,line=2,outer=TRUE)
> par(opar)
```

or else in the set of all pairwise log ratios (Fig. 5.19)

```
> opar <- par(mfrow=c(4,4),mar=c(2,2,1,1), oma=c(4,4,0,0))
> for(i in 1:4){
+  for(j in 1:4){
+   plot(log(Pred[,i]/Pred[,j]),log(Resid[,i]/Resid[,j]) ,
                                    pch=ifelse(i!=j,19,""))
+   if(i==j){text(x=0,y=0,labels=colnames(Resid)[i],cex=1.5)}
+   else{abline(h=0)}
+  }
+ }
> mtext(text=c("predicted values","residuals"),
                  side=c(1,2),at=0.5,line=2,outer=TRUE)
> par(opar)
```

In all these scatterplots, we will never see a linear dependence, because it was removed by the linear model. However, we can see nonlinear structures in the residuals, e.g., U-shaped residual clouds (positive residuals for small prediction, negative for medium, and positive again for high prediction), which would suggest

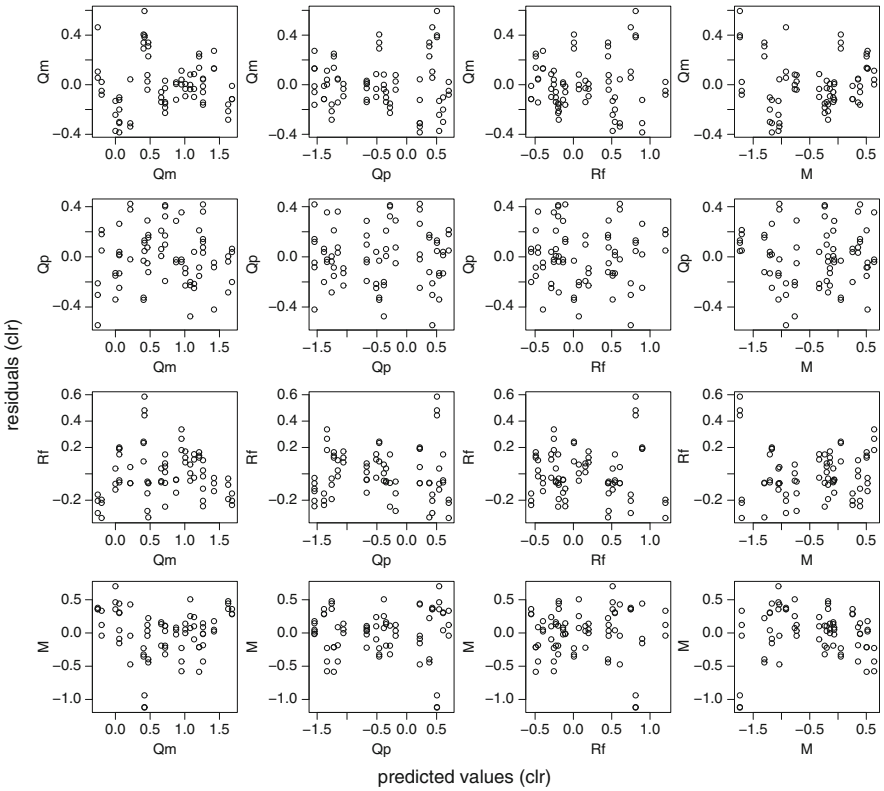


Fig. 5.18 Residuals vs. predicted values of the linear model, expressed in clr coefficients. This diagram helps assessing homoscedastic residuals, i.e., that the spread of all components in the ordinate axis is reasonably homogeneous along each abscissa axis

a nonlinear dependence. We could further observe a different variability of different predicted values (e.g., the diagram involving Q_m and R_f in Fig. 5.19). In this figure, we might observe a structure of the prediction into three groups, probably due to the three different levels of one of the factors (grain size). We could check this by adding color to the plot (not shown here):

```
> opar <- par(oma=c(0,0,2,0),mar=c(4,4,1,0))
> pairwisePlot(clr(Pred),clr(Resid),col=as.numeric(X3))
> legend(locator(1),levels(X3),col=as.numeric(1:3),pch=1,xpd=NA)
> par(opar)
```

According to that plot, the group with the highest variability is the coarse-grain group. This is not surprising, since in a set of coarse grains, we would expect a higher variability for the same volume of sand due to a lower mixing effect. It is left to the reader to obtain the figure.

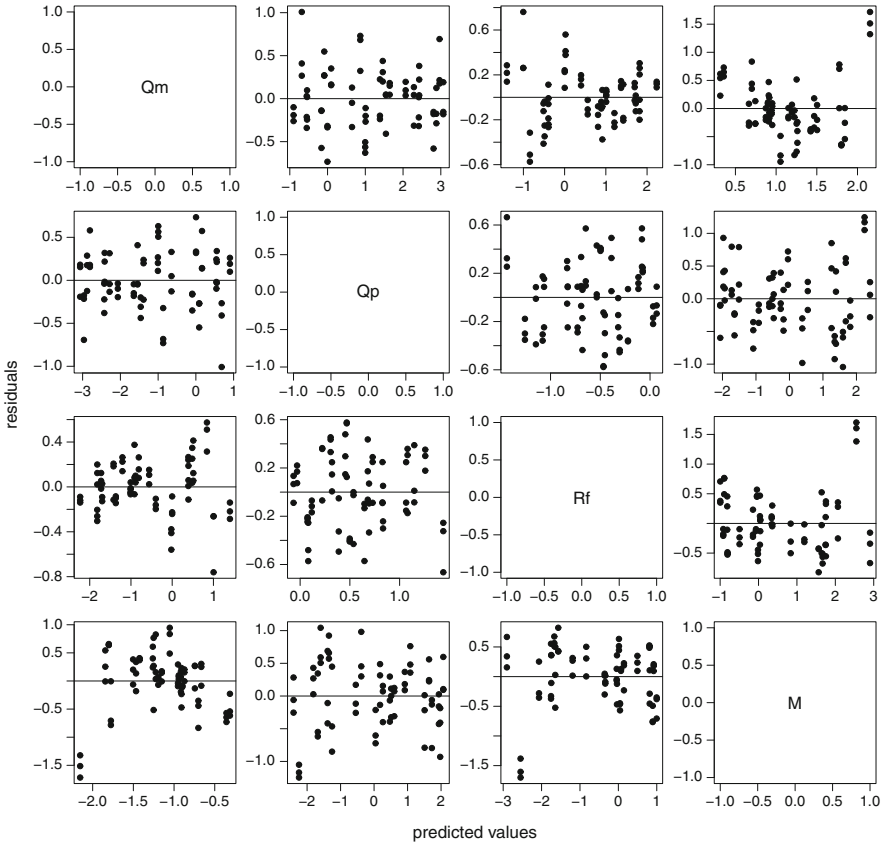


Fig. 5.19 Residuals vs. predicted values of the linear model, in a pairwise log-ratio representation: each diagram represents the ratio of the variable in the row divided by the variable in the column. This diagram helps assessing homoscedastic residuals, i.e., that the spread of all components in the ordinate axis is reasonably homogeneous along each abscissa axis

5.3.9 Measures of Compositional Determination

In classical regression analysis, we use the correlation between explanatory and dependent variables ($\mathrm{cor}(Y, X)$ or its square $R^2 = \mathrm{cor}(Y, X)^2$) to describe the strength of the dependence or how well the regression model describes the observations:

$$R^2 = \frac{\mathrm{var}(\hat{Y}_i)}{\mathrm{var}(Y_i)} = 1 - \frac{\mathrm{var}(\hat{Y}_i - Y_i)}{\mathrm{var}(Y_i)}$$

The advantage of R^2 over the correlation is that it can be computed independently of the form of the model. However, in this equation, we still rely on Y to be univariate.

R^2 can be interpreted as the portion of variance that is removed in the conditional distribution of Y given X by using the linear model rather than an overall mean. We can generalize this further for multivariate linear models by replacing the variance by the metric variance:

$$R^2 = \frac{\text{mvar}(\hat{\mathbf{Y}}_i)}{\text{mvar}(\mathbf{Y}_i)} = 1 - \frac{\text{mvar}(\hat{\mathbf{Y}}_i \ominus \mathbf{Y}_i)}{\text{mvar}(\mathbf{Y}_i)}$$

Although the metric variance might be meaningless for ordinary multivariate linear models, where the variances of different variables may not be comparable, a common variance is a meaningful and sensible concept for compositions (Sect. 4.1), and we can thus safely use it here.

However, there is an additional problem. If R^2 is estimated from data

$$\widehat{\text{mvar}}(\hat{\mathbf{Y}}_i \ominus \mathbf{Y}_i) = \frac{1}{n-1} \sum_{i=1}^{n} d_A^2(\hat{\mathbf{Y}}_i, \mathbf{Y}_i),$$

with the Aitchison distance (2.5), then we get a biased estimator of the residual variance, because we had to estimate all model parameters (not only the mean, for which the -1 usually stands for). We should then adjust the denominator according to the degrees of freedom to get an unbiased estimator:

$$R^2_{\text{adj}} = 1 - \frac{((n-1)/df.\text{residuals}) \cdot \widehat{\text{mvar}}(\hat{\mathbf{Y}}_i \ominus \mathbf{Y}_i)}{\hat{\text{mvar}}(\mathbf{Y}_i)}.$$

Without this correction term, models with more parameters would systematically yield larger R^2, but now R_{adj} can be smaller than 0, which might hinder the interpretation. Function `R2` computes the adjusted coefficient of determination using the metric variance.

When comparing two models, we can interpret that the one with a higher R^2_{adj} value is better in the sense that it provides a more precise conditional expectation. However, this is only an exploratory tool: to check that a parameter really influences the outcome, one should always use a test.

Let us compute R^2_{adj} for some models:

```
> R2(lm(ilr(Y)~log(X1)+X2))

[1] 0.05371

> R2(lm(ilr(Y)~log(X1)+X3))

[1] 0.7749

> R2(lm(ilr(Y)~log(X1)+X2+X4))

[1] 0.08222
```

```
> R2(lm(ilr(Y)~log(X1)+X2+X3))
[1] 0.8103
> R2(lm(ilr(Y)~X2+X3+X4))
[1] 0.7876
> R2(lm(ilr(Y)~log(X1)+X2+X3+X4))
[1] 0.8511
```

These values show a higher degree of determination for the largest model, the one with all variables. But we also get very good models just with grain size (X3) and one of the two continuous variables, log(X1) or X2.

5.4 Compositions as Both Dependent and Independent Variables

Up to here, we have introduced compositions either as the explanatory vector or as the explained one. It is not difficult to have different compositions playing both roles. We may even devise a situation where several compositions are used to predict another composition. For these cases, we just take each composition expressed in their ilr coordinates and treat them as usual, within the framework of a model with several main effects (page 137). We must just take care to back-transform the right things with the right `ilrInv`. Always giving the accessory argument `orig` might help a lot in keeping order in the coefficients or in the predictions.

5.4.1 Example

To demonstrate the use of a second composition as a regressor, we will use the same petrographic composition data, but in a slightly different way. In fact, Grantham and Velbel (1988) took sediment samples from the chosen locations; split them in coarse, medium, and fine sand sediment fractions; and analyzed each fraction for the four components presented in Sect. 5.3.1. Thus, for each observation on coarse grains, there are corresponding observations on medium and fine grains on the same measurement locations and in the same sequence in the dataset.

```
> compY <- acomp(Y[GeoChemSed$grain=="c",])
> compX <- acomp(Y[GeoChemSed$grain=="m",])
> dim(compY)
[1] 24  4
> dim(compX)
[1] 24  4
```

5.4.2 Visualization

Our goal will be now to predict the composition **Y** of the coarse-grain fraction from the petrographic composition of the medium-grain fraction **X**. The rationale behind the model is that the composition of measurements at the same location should be similar. To visualize the relationship, we might plot each of the coordinates or log ratios (clr, alr, ilr, or the set of pairwise log ratios) of **X** in the abscissa axis against **Y** transformed in the same way. That can be done with the `pairwisePlot` command (Fig. 5.20):

```
> opar<-par(mar=c(4,4,0,0),oma=c(3,3,0.1,0.1))
> pairwisePlot(clr(compX),clr(compY))
> mtext(text=c("medium-sized","coarse"),side=c(1,2),
                                  at=0.5,line=2,outer=TRUE)
```

A sort of dependence is visible, especially in the plots on the diagonal.

5.4.3 The Composition-to-Composition Regression Model

In this case, our linear model would be a full-rank transformation of the sediment composition when "passing" from medium to coarse size:

$$\text{ilr}(\mathbf{Y}) = \mathbf{a} + \text{ilr}(\mathbf{X}) \cdot \mathbf{B} + \varepsilon_i$$

where now **B** is a square matrix representing a linear transformation between compositions in ilr space, like usual matrices represent linear mappings of vectors. The model can thus be directly computed and tested in **R**:

```
> (modelCC <- lm(ilr(compY)~ilr(compX)))

Call:
lm(formula = ilr(compY) ~ ilr(compX))
Coefficients:
             [,1]    [,2]    [,3]
(Intercept)  1.1408  0.8130  0.4424
ilr(compX)1  0.8577  0.3372  1.3938
ilr(compX)2  0.4850  0.9173  0.6260
ilr(compX)3  0.2768  0.0829  1.8573

> anova(modelCC)

Analysis of Variance Table
```

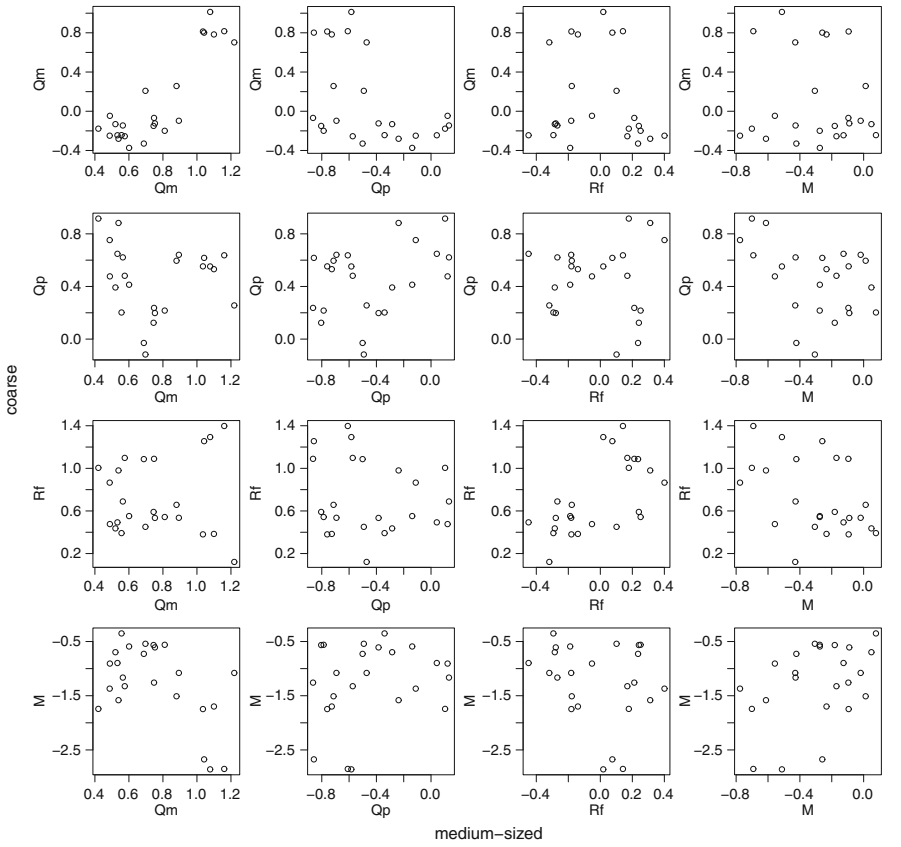


Fig. 5.20 Scatterplots of the coarse sediment composition against the medium-sized sediment composition, both represented in the same set of clr coefficients

```
            Df Pillai approx F num Df den Df  Pr(>F)
(Intercept)  1  0.898     52.8      3     18 4.1e-09
ilr(compX)   3  1.450      6.2      9     60 3.6e-06
Residuals   20
```

Thus, because the p-value is under the critical level of 0.05, we can take the dependence of the $\mathbf{Y}$ on $\mathbf{X}$ as credible. The intercept $\mathbf{a}$ can be interpreted as the ilr transformation of the mean composition of $\mathbf{Y}$ that would correspond to $\mathbf{X} = \mathbb{1}$:

```
> ilrInv(coef(modelCC)[1,],orig=compY)

     Qm      Qp      Rf       M
0.05785 0.29035 0.35078 0.30103
attr(,"class")
[1] acomp
```

5.4.4 Representation and Visualization of Coefficients

On the other hand, the rest of coefficients form the entries of matrix **B**, originally represented in a "double" ilr space. We can apply to this matrix a SVD to understand its structure, like in page 137.

```
> (Bsvd <- svd( ilrvar2clr(coef(modelCC)[-1,]) ))

$d
[1] 2.617e+00 8.967e-01 3.722e-01 1.244e-16

$u
         [,1]     [,2]    [,3] [,4]
[1,] -0.79044  0.27036 -0.2283  0.5
[2,]  0.09674  0.08148  0.8567  0.5
[3,]  0.09662 -0.83104 -0.2237  0.5
[4,]  0.59708  0.47920 -0.4048  0.5


$v
         [,1]    [,2]    [,3] [,4]
[1,] -0.60073  0.5217 -0.3419  0.5
[2,] -0.11202 -0.0462  0.8575  0.5
[3,] -0.07523 -0.7912 -0.3440  0.5
[4,]  0.78798  0.3157 -0.1716  0.5

> opar <- par(xpd=NA,mar=c(2,2,0,1))
> coloredBiplot(Bsvd,xarrows=TRUE,xnames=colnames(Y),
                                        ynames=colnames(Y))
> par(opar)
```

This time, the particularity is that both left and right singular vectors can be interpreted as a basis of the same simplex, one (the left) when this space is the origin of the linear application and the other (the right) when the space is the image of the application. Recall that the function `ilrvar2clr` transforms a square matrix expressed in ilr form to its equivalent expressed in clr, more practical to represent results in plots. We can obtain the vectors of these two bases with

```
> u = clrInv(t(Bsvd$u))
> v = clrInv(t(Bsvd$v))
>  colnames(u) <- colnames(v) <- colnames(Y)
> print(u)

     Qm     Qp     Rf      M
[1,] 0.1014 0.2462 0.24622 0.4061
[2,] 0.2948 0.2440 0.09798 0.3632
[3,] 0.1723 0.5101 0.17314 0.1445
```

```
[4,] 0.2500 0.2500 0.25000 0.2500
attr(,"class")
[1] acomp
```

```
> print(v)
```

```
     Qm     Qp     Rf     M
[1,] 0.1200 0.1957 0.2030 0.4813
[2,] 0.3774 0.2139 0.1015 0.3071
[3,] 0.1538 0.5104 0.1535 0.1824
[4,] 0.2500 0.2500 0.2500 0.2500
attr(,"class")
[1] acomp
```

Note that in both cases, the last row-vector will always be the neutral element of the simplex, because we did the computations with the clr vectors: this last vector should thus be ignored.

Regarding this particular illustration case, image and origin directions are quite similar. This leads to the interpretation that any variation of the $\mathbf{X}$ composition induces a similar change for $\mathbf{Y}$. That coincidence can also be seen in the biplot (Fig. 5.21). As will be explained in Sect. 6.1.2, in a compositional biplot, no ray of a component should be interpreted: our attention should be focused on the links between components (of the same composition, in this case). On the other hand, because this is the biplot of a linear model, an origin link projected onto an image link gives the expected change in the explained composition by a one-unit change in the explanatory composition. For instance, if we take the origin link *QpRf* and project it onto its equivalent image link, we see that the origin is approximately twice as large as the image: thus, a one-unit increment in the $\ln(Qp/Rf)$ of a medium sand composition generates a two-unit increment in the same log ratio in the coarse sand. Since the configurations of all variables in both compositions are quite similar, we can deduce that this is going to be the general rule: changes in the composition of the medium sand are amplified by a factor of two in the coarse sand:

$$\frac{\left(\frac{Qp}{Rf}\right)^{\text{final}}_{\text{coarse}}}{\left(\frac{Qp}{Rf}\right)^{\text{initial}}_{\text{coarse}}} = \left[\frac{\left(\frac{Qp}{Rf}\right)^{\text{final}}_{\text{medium}}}{\left(\frac{Qp}{Rf}\right)^{\text{initial}}_{\text{medium}}}\right]^2$$

This is telling us that under the same environmental conditions, coarse-grained sands have more variability than medium-grained sands. This difference on variances has already been observed discussing homoscedasticity in Sect. 5.3.8.4. That is barely a surprise, as medium sands are actually obtained by comminuting, triturating coarser sands, and mixing the results. They are thus a sort of "average" and should have less variance.

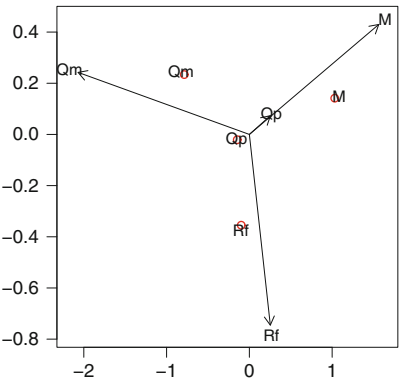


Fig. 5.21 Biplot representation of the linear model of regression of the coarse sand composition (*circles*) as a function of the medium sand composition (*arrows*). Recall that only links between variables should be interpreted

5.4.4.1 Scalar Composition-to-Composition Model

It is generally not recommended to work with the standard `summary` of linear models involving compositions, because it gives an excessive importance to the ilr basis used in the computations, being an arbitrary choice. However, in the case of an auto-regression within the same composition, like in this example, we *can* interpret the tests for individual variables:

```
> summary(modelCC)

Response Y1 :

Call:
lm(formula = Y1 ~ ilr(compX))

Residuals:
    Min      1Q  Median      3Q     Max
-0.5513 -0.0923 -0.0095  0.0995  0.4607

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.141      0.155    7.35  4.2e-07
ilr(compX)1    0.858      0.127    6.75  1.4e-06
ilr(compX)2    0.485      0.183    2.65    0.015
ilr(compX)3    0.277      0.164    1.68    0.108

Residual standard error: 0.207 on 20 degrees of freedom
Multiple R-squared: 0.701,  Adjusted R-squared: 0.656
F-statistic: 15.6 on 3 and 20 DF,  p-value: 1.81e-05
```

```
Response Y2 :

Call:
lm(formula = Y2 ~ ilr(compX))

Residuals:
    Min      1Q  Median      3Q     Max
-0.3848 -0.1437 -0.0281  0.1571  0.4380

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8130     0.1687    4.82  0.00010
ilr(compX)1   0.3372     0.1381    2.44  0.02406
ilr(compX)2   0.9173     0.1988    4.61  0.00017
ilr(compX)3   0.0829     0.1789    0.46  0.64817

Residual standard error: 0.225 on 20 degrees of freedom
Multiple R-squared: 0.551,  Adjusted R-squared: 0.484
F-statistic: 8.18 on 3 and 20 DF,  p-value: 0.000949

Response Y3 :

Call:
lm(formula = Y3 ~ ilr(compX))

Residuals:
    Min      1Q  Median      3Q     Max
-1.0927 -0.3735 -0.0354  0.4059  1.2518

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.442      0.476    0.93   0.3637
ilr(compX)1    1.394      0.390    3.58   0.0019
ilr(compX)2    0.626      0.561    1.12   0.2775
ilr(compX)3    1.857      0.504    3.68   0.0015

Residual standard error: 0.635 on 20 degrees of freedom
Multiple R-squared: 0.498,  Adjusted R-squared: 0.423
F-statistic: 6.62 on 3 and 20 DF,  p-value: 0.00277
```

This tells us that, for each ilr coordinate of the coarse sand, we can simplify the model and use the same ilr coordinate for medium sand as the only significant explanatory variable of the model. This reinforces our interpretation that each change in the medium sand composition has an effect on the same direction for the coarse sand composition. That would lead to a much simpler regression model

of the form

$$\mathbf{Y} = \mathbf{a} \oplus b \odot \mathbf{X}_i + \varepsilon_i$$

where we use a scalar b instead of a full matrix. If we want to obtain this model, we should take coordinates

$$\text{ilr}(\mathbf{Y}) = \text{ilr}(\mathbf{a}) + b\text{ilr}(\mathbf{X}_i) + \text{ilr}(\varepsilon_i).$$

That says that each ilr coordinate may have an own intercept $\text{ilr}(a)$, but they must all have the same slope b. We can then add a fake factor giving the coordinate of each value, destroy the vector structure of the data (with the vectorizer command `c`), and produce a univariate linear model with a continuous and discrete explanatory variables:

```
> (coorindex = factor(rep(1:ncol(ilr(compX)),each=nrow(compX))))

 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
[31] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
[61] 3 3 3 3 3 3 3 3 3 3 3 3
Levels: 1 2 3

> (simplemodel = lm( c(ilr(compY)) ~ c(ilr(compX)) + coorindex ) )

Call:
lm(formula = c(ilr(compY)) ~ c(ilr(compX)) + coorindex)

Coefficients:
  (Intercept)  c(ilr(compX))     coorindex2     coorindex3
        0.998          0.884         -0.503         -2.157
```

which would translate to

$$\text{ilr}(\hat{\mathbf{Y}}) = \big(0.998, 0.998 - 0.503, 0.998 - 2.157\big) + 0.884 \cdot \text{ilr}(\mathbf{X})$$

Note that this model implies that the factor of change b is independent of the direction of change and that the error structure, i.e., the compositional variance of ε_i, is rotation invariant, a sphere on the simplex. In summary, though being apparently very simple, this kind of models is mathematically complex, difficult to handle in **R**, and of very limited flexibility when modeling.

5.5 Advanced Considerations

5.5.1 Interaction Models

In a next step, we could ask whether the influences of different covariables are influenced by each other. The most complex model we have up to now would contain a main effect for each covariable:

$$\begin{aligned}\mathbf{Y}_i &= X1_i \odot \mathbf{b}1 \oplus X2_i \odot \mathbf{b}2 \oplus \mathbf{b}3_{X3_i} \oplus \mathbf{b}4_{X4_i} \oplus \ldots \\ &= X1_i \odot \mathbf{b}1 \oplus X2_i \odot \mathbf{b}2 \oplus \\ &\quad \oplus \delta_{X3_i\text{"}medium\text{"}} \odot \mathbf{b}3_{\text{"}medium\text{"}} \\ &\quad \oplus \delta_{X3_i\text{"}coarse\text{"}} \odot \mathbf{b}3_{\text{"}coarse\text{"}} \\ &\quad \oplus \delta_{X4_i\text{"}south\text{"}} \odot \mathbf{b}4_{\text{"}south\text{"}}\end{aligned}$$

where δ_{xy} is the Kronecker delta function. But each of the covariables can change the influence of another covariable. This can happen in the form of a linear regression or as an analysis of variance and produces additional additive terms to the model equation, which are called *interactions*:

- For two real covariables $X1$ and $X2$ that would be

$$\mathbf{Y}_i = X1_i \odot (\mathbf{b}1 \oplus X2_i \odot \mathbf{b}12') \oplus X2_i \odot (\mathbf{b}2 \oplus X1_i \odot \mathbf{b}21') \oplus \ldots$$

  which can be reformulated to

$$\begin{aligned}\mathbf{Y}_i &= X1_i \odot \mathbf{b}1 \oplus X1_i \odot X2_i \odot \mathbf{b}12' \oplus X2_i \odot \mathbf{b}2 + X2_i \odot X1_i \odot \mathbf{b}21' \oplus \ldots \\ &= X1_i \odot \mathbf{b}1 \oplus X2_i \odot \mathbf{b}2 \oplus (X1_i \; X2_i) \underbrace{(\mathbf{b}12' \oplus \mathbf{b}21')}_{\mathbf{b}12} \oplus \ldots\end{aligned}$$

  where $\mathbf{b}12$ is another compositional parameter and $X1 \; X2$ is just the product of the two explanatory variables. The model becomes then pretty similar to the quadratic regression model (Sect. 5.2.14).
- For two discrete covariables $X3$ and $X4$, we may have one influence parameter for each combination of their levels. Take for illustration the variables `grain` and `position`: we would have then a table of interactions like

fine:north	medium:north	coarse:north
fine:south	medium:south	coarse:south

  But, as happens with the main effects in ANOVA models, the table is not unique, and we must choose a contrast transformation of the table to ensure uniqueness. The **R**-default is to define each of the terms as identically zero for the first combination of categories: e.g., the coefficients in the first row and the first column of the table would be considered zero, and we would estimate only interactions for `medium:south` and `coarse:south`. In the illustration example, the model would finally read

$$\begin{aligned}\mathbf{Y}_i = \ldots & \delta_{X3_i\text{"}medium\text{"}} \odot \mathbf{b}3_{\text{"}medium\text{"}} \\ & \oplus \delta_{X3_i\text{"}coarse\text{"}} * \mathbf{b}3_{\text{"}coarse\text{"}} \\ & \oplus \delta_{X4_i\text{"}south\text{"}} \odot \mathbf{b}4_{\text{"}south\text{"}} \\ & \oplus \delta_{X3_i\text{"}medium\text{"}} \delta_{X4_i\text{"}south\text{"}} \odot \mathbf{b}34_{\text{"}medium\text{"},\text{"}south\text{"}} \\ & \oplus \delta_{X3_i\text{"}coarse\text{"}} \delta_{X4_i\text{"}south\text{"}} \odot \mathbf{b}34_{\text{"}coarse\text{"},\text{"}south\text{"}} \oplus \ldots\end{aligned}$$

- If a categorical covariable modifies the influence of a continuous one, we get different intercepts and slopes of the continuous covariable for each level of the categorical one. In the illustration case, for instance,

$$\begin{aligned}\mathbf{Y}_i &= \mathbf{a}(X3_i) \oplus X1_i \odot \mathbf{b}1_{X3_i} \oplus \cdots \\ &= (\mathbf{a} \oplus \delta_{X3_i\text{"}medium\text{"}} \odot \mathbf{a}3_{\text{"}medium\text{"}} \oplus \delta_{X3_i\text{"}coarse\text{"}} \odot \mathbf{a}3_{\text{"}coarse\text{"}}) \oplus \\ &\quad \oplus X1 \odot (\mathbf{b}1 \oplus \delta_{X3_i\text{"}medium\text{"}} \odot \mathbf{b}13_{\text{"}medium\text{"}} \oplus \delta_{X3_i\text{"}coarse\text{"}} \odot \mathbf{b}13_{\text{"}coarse\text{"}}) \oplus \cdots,\end{aligned}$$

  which again uses the treatment contrasts, taking zero interactions with the first category. Thus, the model would be equivalent to three regression lines, one for each grain size, with intercepts and slopes: $\mathbf{a}$ resp. $\mathbf{b}$ for fine sand; $\mathbf{a} \oplus \mathbf{b}3_{\text{"}medium\text{"}}$ resp. $\mathbf{b} \oplus \mathbf{b}13_{\text{"}medium\text{"}}$ for medium sand; and $\mathbf{a} \oplus \mathbf{b}3_{\text{"}coarse\text{"}}$ resp. $\mathbf{b} \oplus \mathbf{b}13_{\text{"}coarse\text{"}}$ for coarse sand.

In all three cases, we get additional compositional parameters following the same structure as the usual main effect terms of linear models. An ilr or alr transformation of the whole model yields a full, classical multivariate linear model with interaction terms.

Classical linear models in **R** admit interactions, marked with the symbols $*$ and : in the model formula:

```
> GeoChemSed=read.csv("GraVel.csv",header=TRUE)
> Y=acomp(GeoChemSed[,7:10])
> names(GeoChemSed)

 [1] "watershed" "position"  "CCWI"      "discharge" "relief"
 [6] "grain"     "Qm"        "Qp"        "Rf"        "M"

> Covariables=GeoChemSed[,c("discharge","relief","grain","position")]
> X1 = Covariables$discharge
> X2 = Covariables$relief
> X4  = factor(Covariables$position)
> X3  = factor(Covariables$grain,c("f","m","c"),ordered=TRUE)
> levels(X3) <- c("fine","medium","coarse")
> contrasts(X3)<-"contr.treatment"
> anova(lm(ilr(Y)~log(X1)+ X2 + X3 + X4 + log(X1)*X2+X2*X3+X3*X4))
```

```
Analysis of Variance Table

            Df Pillai approx F num Df den Df  Pr(>F)
(Intercept)  1  0.944      330      3     59 < 2e-16
log(X1)      1  0.667       39      3     59 4.1e-14
X2           1  0.795       76      3     59 < 2e-16
X3           2  1.196       30      6    120 < 2e-16
X4           1  0.486       19      3     59 1.3e-08
log(X1):X2   1  0.253        7      3     59 0.00059
X2:X3        2  0.175        2      6    120 0.08294
X3:X4        2  0.604        9      6    120 7.8e-08
Residuals   61
```

In general, `X1:X2` represents the interaction of the two variables, whereas `X1*X2` represents the interaction and the main effects, `X1*X2=X1+X2+X1:X2`. Thus, it is also possible to omit the terms for the main effects, since they are always implicitly present, when the interaction terms are specified as products. The following shorter description would give the same result:

```
> anova(lm(ilr(Y)~log(X1)*X2+X2*X3+X3*X4))
```

If we decide that the interaction term `X2:X3` of $X2$ and $X3$ is no longer significant at the 5 % level, we can omit it:

```
> anova(lm(ilr(Y)~log(X1)*X2+X3*X4))
```

Further higher-order terms can be added by longer products, like in

```
> anova(lm(ilr(Y)~log(X1)*X2+X2*X4+log(X1)*X3*X4))

Analysis of Variance Table

              Df Pillai approx F num Df den Df  Pr(>F)
(Intercept)    1  0.976      759      3     55 < 2e-16
log(X1)        1  0.714       46      3     55 5.6e-15
X2             1  0.807       77      3     55 < 2e-16
X4             1  0.739       52      3     55 4.9e-16
X3             2  1.324       37      6    112 < 2e-16
log(X1):X2     1  0.273        7      3     55  0.0005
X2:X4          1  0.491       18      3     55 3.7e-08
log(X1):X3     2  0.252        3      6    112  0.0177
log(X1):X4     1  0.102        2      3     55  0.1124
X4:X3          2  0.814       13      6    112 5.7e-11
log(X1):X4:X3  2  0.462        6      6    112 4.0e-05
Residuals     57
```

In this case, a three-order term is significant, whereas one of its nested second-order terms is no longer so. To ensure that the term as a whole is necessary, we would need to check what happens with the significance of the coefficients, if the third-order

interaction is added together with or without its enclosed second-order interactions, by explicitly adding/removing the specific terms:

```
> anova(lm(ilr(Y)~log(X1)*X2+X2*X4+log(X1):X3:X4))

Analysis of Variance Table

              Df Pillai approx F num Df den Df  Pr(>F)
(Intercept)    1  0.958      451      3     59 < 2e-16
log(X1)        1  0.629       33      3     59 1.0e-12
X2             1  0.798       77      3     59 < 2e-16
X4             1  0.527       22      3     59 1.2e-09
log(X1):X2     1  0.245        6      3     59 0.00079
X2:X4          1  0.339       10      3     59 1.8e-05
log(X1):X4:X3  5  1.625       14     15    183 < 2e-16
Residuals     61
```

All terms in this model are now highly significant.

5.5.2 Akaike Information Criterion

Another general criterion often proposed for model selection is the *Akaike information criterion* (*AIC*) (Sakamoto et al., 1986), given by

$$AIC(\text{model}) = 2 * p - \sum_{i=1}^{n} 2 \ln(f_{\hat{\theta}}(y_i | X_i = x_i))$$

where θ is the ML estimate of the parameter y_i, $f_{\hat{\theta}}(y|X = x)$ the conditional density of the dependent variable given the covariables, and p is the dimension of the parameter. Here entropy is defined with respect to the Aitchison measure of the simplex. The AIC can have different values with different definitions of whether or not the variance parameter needs to be counted in p and what reference measure is used for the density functions. However, for fixed choices and a single dataset, the values are always comparable. It is thus possible to use $p =$ `length(coef(model))`, which is what we get if we compute the criterion with the `AIC` command from **R**. However, at the time of writing this book, the command gives "inadequate" results for multivariate linear models, because it implicitly uses a `logLik.lm` which is unaware of the subtle differences between simple linear regression and multivariate linear models.

```
> AIC(model)

[1] 93.04

> AIC(lm(ilr(Y)~log(X1)+X3))
```

```
[1] 184.7
> AIC(lm(ilr(Y)~log(X1)+X2+X3+X4))
[1] 93.04
> AIC(lm(ilr(Y)~log(X1)*X3))
[1] 185.7
> AIC(lm(ilr(Y)~log(X1)*X2*X3*X4))
[1] -201.3
> AIC(lm(ilr(Y)~log(X1)*X2*X3+X1*X3*X4))
[1] -181.7
```

Smaller *AIC* correspond to the worse models, either because their likelihood is too small or because they use too many parameters. The correction term $2p$ thus plays the same role as the adjustment in R^2: a larger model always has a larger likelihood, thus they have to be penalized somehow. Alternative corrections have been proposed (e.g., the Schwarz *Bayesian Information Criterion*, *BIC*) and are also available through `AIC`. However, the generalization to the compositional situation has not been fully explored.

5.5.3 Model Selection

5.5.3.1 Some General Thoughts on Model Selection

The general strategies for selecting the right one from a set of linear models are unaffected by the compositional nature of the problem. Several techniques like the lattice of hypothesis, forward selection, backward selection, combined forward and backward selection, and the use of AIC, BIC, or R^2 have been proposed in the literature. It is not our aim to choose the best of these strategies. We will just show how to use each of them in **R**. It is however useful to gather some basic ideas on the general aim and scope of all selection criteria.

- Models should be as simple as possible and as complicated as necessary. Significant tests or logic evidence prove necessity. The number of parameters and scientific interpretability measure simplicity.
- No statistical procedure can identify the one true model. Rigorous testing approaches can at most just prove that a good model does (at least) contain the proven terms.
- No model is true; some are useful. What is useful depends on the question at hand.
- There are very many models, too many to look at them, and probably many of them will provide a similar degree of determination.

- If a model does not consider known relevant covariables, the residuals will not be independent for variables with similar covariables. Covariables known to be relevant should thus be included, even if not significant.
- Nonsignificant parameters do not improve prediction, even if they are known to be relevant.
- The theory as presented is not robust, because we are not aware of a robust estimation theory for multivariate linear models at a workable and implemented level in **R**.
- The whole theory of linear models is based on the assumptions of homoscedasticity and normality, which is here transformed to additive logistic normality (normality in the simplex). Due to the Aitchison geometry of the simplex, both assumptions are possible and reasonable, although (just like in usual regression) probably never exactly fulfilled.
- This book provides a frequentist's view of the theory, which is only for reasons of availability in the basic **R**. A proper Bayesian approach will always be superior, when realistic prior information is available. There are several packages implementing Bayesian methods that can be just plugged into a compositional analysis following the lines shown here with linear models.

5.5.3.2 Obtaining All Possible Models

We have clearly seen in the last section that the number of alternative models quickly becomes too large for a manual exploration, when we consider interactions between covariables. The same happens if we suspect that only a subcomposition of our compositional variables does actually show a dependence, and we want to extract it by exploring all possible subcompositions. If just with $P = 4$ covariables we found an impractical 32,768 of alternative models, when exploring subcompositions, things are not that terrible. With D parts, we can build:

- Subcompositions of two parts, of three parts, etc.
- Up to one "subcomposition" of D parts (having all elements)

These are the elements of the Dth row of Tartaglia's triangle, except the first two because a subcomposition of zero elements is nothing and a subcomposition of one element has no sense. For instance, with $D = 4$, we would get $6 + 4 + 1 = 11$ subcompositions to explore. In any case, to select covariables and/or subcompositions to explore, we can intensively use the `lapply` and `sapply` functions, devised to "apply" a given function to the elements of a vector or a list.

Take, for instance, the exploration of all second-order interactions, with and without their main effects, in the Grantham and Velbel (1988) dataset (for illustration, we consider now relief as not significant). We must first take the global model and extract all the possible terms:

```
> maxmodel = lm(ilr(Y)~log(X1)*X3+log(X1)*X4+X3*X4 )
> (terms = rownames(coef(maxmodel))[-1])
```

```
[1] "log(X1)"          "X3medium"         "X3coarse"
[4] "X4south"          "log(X1):X3medium" "log(X1):X3coarse"
[7] "log(X1):X4south"  "X3medium:X4south" "X3coarse:X4south"
```

Before continuing, we must create some new variables, with the names of the terms of the categorical variables (except the first category):

```
> X3medium = X3=="medium"
> X3coarse = X3=="coarse"
> X4south =  X4=="south"
```

The second step is to get all possible subsets of one, two, three, etc. elements, up to the subset of all terms. These are all combinations with any number of terms among those 9, obtained with

```
> idx = sapply(1:length(terms), function(i){combn(terms,i) } )
```

This computes the combinations of the elements in `terms` taken in groups of `i` elements (resulting in a matrix of `i` rows and as many columns as different combinations exist), with `i` looping from 0 to the total number of terms (all stored in a single list of 10 elements).

We must then loop through all elements of this list, and take each column once, to select the terms we want to include in the model. These will be then combined in a formula and fed to a linear model. All these steps must be enclosed in a couple of ad hoc functions inside `sapply` calls:

```
> res = lapply(idx, function(termstaken){
+   tt = termstaken
+   if(is.null(dim(tt)))dim(tt)=c(length(tt),1)
+   lapply(1:ncol(tt), function(icol){
+    frm = paste( tt[,icol], collapse="+")
+    frm = paste("ilr(Y)~", frm, sep="")
+    mylm = lm(as.formula(frm))
+    return(mylm)
+   })
+ })
```

This results in a two-level nested list, containing the results of all linear models, grouped by the length of the subcomposition analyzed. Now we should destroy the first nesting, in order to have all models in the same level, independently of the number of terms included:

```
> length(res)

[1] 9

> length(res[[1]])

[1] 9
```

```
> res = unlist(res, recursive=FALSE)
> length(res)

[1] 511

> names(res) = 1:length(res)
```

It is convenient to give names to the several models, in order to be able to identify them afterwards (this is done in the last line).

5.5.3.3 Choosing the Best Models

At this point, we can apply any model selection criterion to the elements of object `res`. But as was said in Sect. 5.5.2, the current **R** implementation is not good for multivariate linear models. Thus, we may have to build the criterion ourselves:

```
> mylogLik.mlm = function(mylm){
+   x = mylm$residuals
+   detvarx = ifelse(length(dim(x))>1,
+          determinant(var(x))$modulus,
+          log(var(x)))
+   ncolx = ifelse(length(dim(x))>1,ncol(x),1)
+   -0.5*(sum(mahalanobis(x,0*mean(x),var(x)))+detvarx +
                                        ncolx * log(2*pi))
+ }
> lL = sapply(res,mylogLik.mlm)
> lc = sapply(res,function(mylm){length( coef(mylm)) })
> rvc = sapply(res,function(mylm){
+        d = max(ncol(mylm$residuals),1) #$
+        return(d*(d+1)/2)
+   })
> np = lc + rvc
> sort(myAIC <- -2*lL + 2*np)[1:10]

    3     4     2     1    18    25    11     6    14    12
237.6 238.7 238.8 238.9 242.6 243.3 243.4 243.4 243.4 244.4

> sort(myBIC <- -2*lL + log(nrow(Y)) * np)[1:10]

    3     4     2     1    18    25    11     6    14    12
264.9 266.0 266.1 266.2 276.7 277.5 277.5 277.5 277.5 278.5
```

Thus, the best models are the first four, as a difference of up to three units in the information criteria is generally (Simonoff, 2003, e.g.) considered as not informative:

```
> lapply(res[c(3,4,2,1)], coef)
```

```
$`3`
                [,1]    [,2]      [,3]
(Intercept)  -1.347 -0.1859 -0.002099
X3coarseTRUE  1.598  0.5522 -1.459986

$`4`
                [,1]     [,2]    [,3]
(Intercept) -0.5914  0.05274 -0.4363
X4southTRUE -0.4459 -0.10910 -0.1049

$`2`
                  [,1]     [,2]    [,3]
(Intercept)  -0.79910  0.07003 -0.5619
X3mediumTRUE -0.04562 -0.21551  0.2194

$`1`
                [,1]    [,2]    [,3]
(Intercept)  2.5431  1.1077 -5.1759
log(X1)     -0.6956 -0.2299  0.9711
```

All these four models should be considered equally valid. In them, we either consider one grain size, the position, or the discharge (in log scale) as informative enough to explain the observed variability given the scarcity of data available.

5.5.3.4 Selecting the Dependent Subcomposition

With the Grantham and Velbel (1988) dataset, we may also explore all possible subcompositions, with the aim of removing a component or two from the global model response:

```
> maxmodel = lm( ilr(Y)~log(X1)+X2+X3+X4)
```

We have to obtain the 11 possible subcompositions (in this case, we will not take the complete composition):

```
> (idx = lapply(2:(ncol(Y)-1), function(i){combn(ncol(Y),i)}))
```

```
[[1]]
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    2    2    3
[2,]    2    3    4    3    4    4

[[2]]
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    2
[2,]    2    2    3    3
[3,]    3    4    4    4
```

The second step is to loop through all elements of this list and take each column once to select the variables from the compositional response. These will define a special ilr basis of $\mathbb{S}^D$, where the first coordinates are dependent on our explanatory variables, and the remaining are independent, i.e., a constant linear model. All these steps must be enclosed in a couple of ad hoc functions inside `sapply` calls:

```
> res = lapply(idx, function(subcomps){
+   lapply(1:ncol(subcomps), function(icol){
+    idparts = unique(c(subcomps[,icol],1:ncol(Y)))
+    sY = acomp(Y,idparts)
+    Y = ilr(sY)
+    Y1 = Y[,1:(nrow(subcomps)-1)]
+    Y2 = Y[,-(1:(nrow(subcomps)-1))]
+    lm1 = lm( Y1~log(X1)+X2+X3+X4)
+    lm2 = lm( Y2~1 )
+    return( list(lm=lm1,nolm=lm2) )
+   })
+ })
```

This results again in a multiple-level nested list, containing the results of all linear models, grouped by the length of the subcomposition analyzed. The next step is then to destroy the first nesting, in order to have the ten models in the same level, independently of the number of parts of their subcompositions:

```
> length(res)

[1] 2

> length(res[[1]])

[1] 6

> res = unlist(res, recursive=FALSE)
> length(res)

[1] 10

> names(res) = unlist( lapply(idx,function(subcomps){
+       lapply(1:ncol(subcomps),
+          function(icol){
+            paste(colnames(Y)[subcomps[,icol]],collapse="-")
+          }
+       ) } ) )
```

Note that the last line just creates some nice names for the ten simplified models considered.

This time it is quite tricky to apply the model selection criteria, because the model itself has always two parts: some coordinates follow a linear model; some other are constant. We must thus adequately define the functions of computation of log-likelihood and estimated parameters:

```
> mylogLik = function(mylm){
+   x = cbind(mylm$lm$residuals,mylm$nolm$residuals)
+   detvarx = ifelse(length(dim(x))>1,
+             determinant(var(x))$modulus,
+             log(var(x)))
+   ncolx = ifelse(length(dim(x))>1,ncol(x),1)
+   -0.5*(sum(mahalanobis(x,0*mean(x),var(x)))+detvarx +
                                        ncolx * log(2*pi))
+ }
> myPars = function(mylm){
+   x = cbind(mylm$lm$residuals,mylm$nolm$residuals)
+   d = ncol(x)
+   return(d*(d+1)/2 + length( coef(mylm$lm)) +
                               length( coef(mylm$nolm) ))
+ }
> ( lL = sapply(res,mylogLik) )

   Qm-Qp    Qm-Rf     Qm-M    Qp-Rf     Qp-M     Rf-M Qm-Qp-Rf
  -106.6   -106.6   -107.2   -107.1   -107.0   -107.1   -105.9
 Qm-Qp-M  Qm-Rf-M  Qp-Rf-M
  -106.4   -106.4   -106.4

> ( np = sapply(res,myPars) )

   Qm-Qp    Qm-Rf     Qm-M    Qp-Rf     Qp-M     Rf-M Qm-Qp-Rf
      14       14       14       14       14       14       19
 Qm-Qp-M  Qm-Rf-M  Qp-Rf-M
      19       19       19

> sort(myAIC <- -2*lL + 2*np)

   Qm-Qp    Qm-Rf     Qp-M     Rf-M    Qp-Rf     Qm-M Qm-Qp-Rf
   241.2    241.3    241.9    242.1    242.2    242.4    249.9
 Qm-Qp-M  Qp-Rf-M  Qm-Rf-M
   250.7    250.7    250.9

> sort(myBIC <- -2*lL + log(nrow(Y)) * np)

   Qm-Qp    Qm-Rf     Qp-M     Rf-M    Qp-Rf     Qm-M Qm-Qp-Rf
   273.1    273.2    273.8    274.0    274.1    274.3    293.1
 Qm-Qp-M  Qp-Rf-M  Qm-Rf-M
   294.0    294.0    294.1
```

We can also work with the generalized likelihood ratio tests. In this case, it is well known that the log ratio of the likelihoods of the maximal model against each of the simplified models (times -2) is approximately distributed as a χ^2 with degrees of freedom equal to twice the number of *removed* parameters when passing from the maximal to the simplified model:

```
> incrlL = mylogLik.mlm(maxmodel) - lL
> incrnp = ( length(coef(maxmodel)) + 3*4/2 ) - np
> (alpha.levels <- pchisq(-2*incrlL,df=incrnp*2,lower.tail=FALSE))

   Qm-Qp    Qm-Rf     Qm-M    Qp-Rf     Qp-M     Rf-M Qm-Qp-Rf
       1        1        1        1        1        1        1
Qm-Qp-M  Qm-Rf-M  Qp-Rf-M
       1        1        1
```

Result show that the generalized likelihood ratio test has no power in this illustration case, probably due to the small sample size. That is also quite true for the Akaike's and Schwarz' criteria: in all cases, the models built on subcompositions of two parts are as good as the global model.

5.5.4 Outliers and Heavy-Tailed Distributions

5.5.4.1 The Lack of Robust Linear Models

It would be highly desirable to have a good robust theory, where heavy-tailed distributions of errors or the presence of extremes and outliers were taken into account, but prevented of spoiling the efficiency properties of estimators and the correctness of α-levels of tests. However, such methodology is not yet available for multivariate models: in **R**, the standard package for robust linear models `roblm` package (Salibian-Barrera, 2006) does not yet (i.e., March 2013) support multivariate linear models. We thus have no choice, but to work with the linear models we have now. This makes it necessary to understand the effects of deviations from the standard assumptions of linear models. The next subsections explore some of these deviations.

5.5.4.2 Validity of Classical Estimators

The regression parameter estimation (i.e., the coefficients) can be legitimated as unbiased estimators, without any reference to the normality assumption or homoscedasticity. The classical parameter estimators are unbiased, if the expectation of the residuals is zero. Furthermore, if the variance of the error terms exists, then the coefficient estimators have a bounded variance, even if these variance vary. However, variance estimates rely on homoscedasticity and test distributions further on the normality assumptions. Parameter estimation is thus always legitimitate, though we should not trust their uncertainty regions if residual normality or homoscedasticity (i.e., same error variances for all cases) is not credible.

However, we will seldom find clear heteroscedasticity in practical situations, because the empirical variance of a random sample is already a highly variable statistic: variances should be compared with quotients, as they honor a ratio scale.

Thus, significant changes must change the order of magnitude of the variance! In other words, a little bit of variation in the spread should not change the validity of our methods.

Beyond these "small heteroscedasticity" situations, the effect of clearer inhomogeneous variances is quite complex. It is clear that wherever the variation of error variance is large, some of the variance estimators are grossly underestimating their true values. That is particularly true for the prediction variance and the prediction error variance for cases of high error variance.

5.5.4.3 Pondering Multivariate Outliers

It is well known that outliers are particularly harder to find when the models are estimated with non-robust methods. That will be mentioned when discussing outliers in a descriptive analysis (Sect. 7.4), where non-robust means and variances where particularly bad if used to detect outliers.

Outliers can have virtually any distorting effect in linear models. They can artificially increase the variance estimates, making tests too conservative. But they can also strongly modify the local mean of a specific parameter combination, thus shifting the parameter estimate substantially towards the outliers, quite like the classical lever effect (Sect. 5.2.10), which would eventually make the tests too restrictive.

Thus, if we find a parameter which is significant in the presence of an outlier and loses its significance when the outlier is removed, we might claim that the outlier is the only responsible of the significance of this parameter, not the randomness of the model.

On the other hand, if a parameter becomes significant after removing an outlier, then we might be inclined to argue that the outlier artificially increased the residual variance and thus masked the otherwise significant parameter. However, we should also consider the alternative that the suspected outlier is a real observation, carrying counterevidence against the parameter. That might be eventually the case if residuals have a skewed distribution, rendering all large values as outliers. In many cases, classical diagnostic plots (Sect. 5.2.10) should help to understand the effect of the outlier, though they are not (strictly speaking) as useful as in the univariate case.

5.5.4.4 Heavy-Tailed Error Distributions

In the presence of heavy-tailed distributions, the quantiles of the relevant distributions of the test statistics can be substantially changed. It is like having a large portion of a dataset that does not contribute much to the variability, together with some values in the tails, which dominate the distribution, as if we would have less observations. This effect is probably stronger for extreme quantiles. The test distributions will thus get heavy tails too. The significance level of tests and all sort of confidence regions would thus become more unreliable for more extreme α. In

this case, we should only use reasonable α-levels in the interpretation and not trust p-values beyond that level. A simulations study can eventually help to quantify this resulting impression. Bootstrap methods might be another tool to compute correct confidence limits and p-values in the presence of heavy-tailed distribution, though a detailed discussion of this subject is beyond the scope of this book.

References

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics*. New York: Wiley.

Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). *Graphical methods for data analysis*. Pacific Grove: Wadsworth & Brooks/Cole.

Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. New York: Chapman and Hall.

Daunis-i Estadella, J., Egozcue, J. J., & Pawlowsky-Glahn, V. (2002). Least squares regression in the simplex. In U. Bayer, H. Burger, & W. Skala (Eds.), *Proceedings of IAMG'02—The eighth annual conference of the International Association for Mathematical Geology*, Volume I and II (pp. 411–416). Berlin: Selbstverlag der Alfred-Wegener-Stiftung, 1106 pp.

Fahrmeir, L., Hamerle, A., & Tutz, G. (1996). *Multivariate statistische Verfahren*. Berlin: De Gruyter.

Grantham, J. H., & Velbel, M. A. (1988). The influence of climate and topography on rock-fragment abundance in modern fluvial sands of the southern blue ridge mountains, north-Carolina. *Journal of Sedimentary Petrology, 58*, 219–227.

Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Maechler, M., et al. (2007). *robustbase: Basic robust statistics*. R package version 0.2-8.

Sakamoto, Y., Ishiguro, M., & Kitagawa, G. (1986). *Akaike information criterion statistics*. The Netherlands: Springer.

Salibian-Barrera, M. (2006). *roblm: MM-regression estimators*. R package version 0.6.

Simonoff, J. S. (2003). *Analyzing categorical data*. Springer texts in statistics. New York: Springer.

Chapter 6
Multivariate Statistics

Abstract Compositions are multivariate by nature. Several multivariate techniques like principal component analysis, cluster analysis, or discriminant analysis have their compositional counterparts. These are mostly applications of the principle of working in coordinates, though not always in a straightforward fashion. The definition, the corresponding tools, and the compositional interpretation of these exploratory techniques are discussed here.

6.1 Principal Component Analysis: Exploring Codependence

6.1.1 The Singular Value Decomposition

Principal component analysis (PCA) is an interpretation of the singular value decomposition (SVD) of a data matrix. For compositional data, the SVD of a *centered, clr-transformed* dataset $\mathbf{X}^*$ is a product of three matrices:

$$\mathrm{clr}(\mathbf{X}^*) = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{V}^t,$$

where

- $\mathbf{V}$ rows identify the variables; its columns are orthonormal vectors of D components, called *right singular vectors*, *loadings*, or *principal components*; they define an orthonormal basis of the simplex.
- $\mathbf{D}$ is a diagonal matrix, with D positive values in decreasing order called the *singular values*; they are interpreted as the standard deviations of the coordinates of the dataset on the new basis.
- $\mathbf{U}$ rows identify the observations, and its columns are orthonormal vectors of N components, called *left singular vectors* or *scores*; the rows of $\mathbf{U}$ are interpreted as the standardized coordinates of the dataset on the new basis.

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_6, © Springer-Verlag Berlin Heidelberg 2013

Now select the first r singular values and its associated first r left and right singular vectors, and compute with them $\mathbf{X}_r = \mathbf{U}_r \cdot \mathbf{D}_r \cdot \mathbf{V}_r^t$. This happens to be the *best* rank r approximation to the actual $\mathbf{X}^*$ (in a least-squares sense, Eckart and Young, 1936). The overall quality of this approximation can be measured by the proportion of preserved variance:

$$\pi_r = \sum_{i=1}^{r} d_{ii}^2 / \sum_{j=1}^{D} d_{jj}^2. \tag{6.1}$$

The importance of this decomposition lies thus in the fact that it allows an optimal representation of a multidimensional dataset in a lower dimension. In particular, it will be useful to optimally represent a D-part compositional dataset in a 2D plot, the *biplot*. The clr transformation must be applied so that the left and right singular vectors reproduce the relative scale of compositional data. PCA is available in **R** with the command `princomp`, with a method for "`acomp`" datasets. This function returns a `princomp` object, containing the full result of a PCA on the covariance matrix of the clr-transformed dataset. The SVD matrices can be found in the following elements of this object:

- "`loadings`" contains the matrix $\mathbf{V}$.
- "`sdev`" contains a vector with the diagonal elements of $\mathbf{D}$ in decreasing order of magnitude.
- "`scores`" contains the matrix $\mathbf{U}$.

6.1.2 Covariance Biplot as Exploratory Tool

One of the most common uses of SVD/PCA is in exploratory analysis. After looking at the predefined projections of ternary diagrams and ad hoc balance scatterplots in Chap. 4, we can "let the data speak" and try to find some other projections better describing the particular structure of the dataset at hand. A powerful tool to select which projections deserve exploration is the *biplot* (Gabriel, 1971). A biplot is a graphical representation of the case $r = 2$ (or in 3D, one can try an $r = 3$ biplot), where variables and observations are displayed, a sort of simultaneous projection of the data cloud and the variable axes onto a two-dimensional plot (or 3D).

To construct a biplot, we must first choose a suitable value for a shape parameter α, in the interval between 0 and 1 (this last one is the default of **R**). Then, the observations are typically represented as dots, in the positions given by the rows of $\mathbf{U}_r \cdot \mathbf{D}_r^{(1-\alpha)}$. On the other side, the variables are commonly plotted as arrows from the center of the plot to the rows of $\mathbf{V}_r \cdot \mathbf{D}_r^{\alpha}$. Biplots for $\alpha = 0$ are called *form biplot*, whereas those for $\alpha = 1$ are *covariance biplot* .

In a non-compositional, covariance biplot, the arrow lengths are proportional to the variances of each variable, and the cosine of the angle between any two arrows is

the correlation coefficient between those two variables. For compositional data, one must remember that the SVD was applied to the clr-transformed data. This implies that we cannot directly interpret the rays of a compositional biplot: each represents the clr variance of a part and has a quite complex relationship with *all* the original parts. Instead, we can look at the links between arrow heads, which represent the log ratios between the two involved parts and are intimately connected with the variation matrix (4.3) of Chap. 4. The following rules may be taken into account when using a covariance biplot to explore the compositional structure of dependence:

1. Two proportional variables have a quasi-constant log ratio (a small entry in the variation array); thus their link should be very short, and the arrow heads lie together.
2. Inversely, if a link is very long, the log ratio of the two parts involved is highly variant (a large entry in the variation array). If we see three very long rays pointing towards different directions (at 120° approximately), a ternary diagram of these three parts will have a high spread, because their mutual links are also very long.
3. The angle between two links should approximate the correlation coefficient between the two log ratios:
   - Two uncorrelated log ratios provide orthogonal rays.
   - Three or more parts lying on a common line have links forming either 0° or 180°, thus being perfectly correlated (directly or inversely); in this case the subcomposition formed by these parts should show a one-dimensional pattern of variation; i.e., this subcomposition is probably *collinear*.
   - Two sets of collinear subcompositions which rays form 90° are (possibly) uncorrelated.

It is of extreme importance to keep in mind that the biplot is as good as the proportion of the total variance actually displayed by the biplot (Example 6.1 shows how to compute that variance proportion). Any conclusion drawn should be further contrasted by plotting the suggested ternary diagrams and log-ratio scatterplots. In the end, the biplot is just an exploratory tool, yet a powerful one to uncover relations between many variables.

Example 6.1 (Obtaining and interpreting a compositional biplot). The first step to obtain a biplot is to produce and store a PCA, available through the generic function `princomp(x)`, with an `acomp` method. This function returns a `princomp` object, containing the full result of a PCA on the covariance matrix of the clr-transformed dataset:

```
> GeoChemSed=read.csv("geochemsed.csv",header=TRUE,skip=1)
> x = acomp(GeoChemSed,4:13)
> pcx = princomp(x)
```

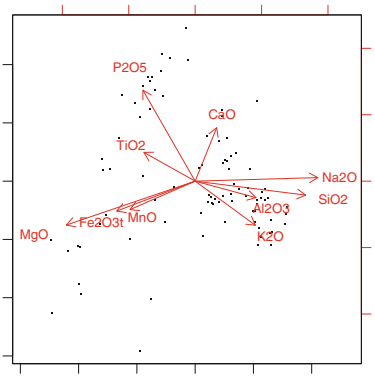


Fig. 6.1 Example of a compositional biplot

The first thing we should do is to obtain the proportion of explained variance that the biplot will capture. This is obtained as

```
> sum(pcx$sdev[1:2]^2)/mvar(x)
[1] 0.9026
```

quite a good value. The biplot of the data is obtained with

```
> opar <- par(mar=c(1,1,1,1))
> dots = rep(".",times=nrow(x))
> biplot(pcx, xlabs=dots)
> par(opar)
```

or `plot(pcx, xlabs, type="biplot")`. Here, the optional argument `xlabs` is used to get biplots where the data are just represented by a dot, instead of labeled with a number (the default, quite noisy for large datasets). What can be seen in such a biplot?

First, some short links are easy to spot, leading to quite low-variance subcompositions: Al–Na–Si from one side (main constituents of the so-called *felsic* minerals, quartz and feldspar) and Fe–Mg–Mn on the other (from the *mafic* minerals, like biotite and garnet). If we compare the metric variances of these subcompositions with the total metric variance

```
> mvar( acomp(x[,c("Al2O3","Na2O","SiO2")] ) )
[1] 0.08767
> mvar( acomp(x[,c("Fe2O3t","MgO","MnO")] ) )
[1] 0.06577
```

```
> mvar(x)

[1] 1.541
```

it is evident that the parts in each one of these groups have very low variances (respectively, 5.7 % and 4.3 % of the total variance); thus they are quite proportional.

But the biplot is also a powerful tool to identify possible one-dimensional patterns, on those subcompositions lying on a common link: for instance, Mn–Fe–K, P–Ca–Na, or P–Ti–Mn may satisfy this condition, as can be checked in Fig. 6.2. This means that we can find a projection of the selected variables onto a particular direction that is quasi-constant. This direction is provided by the **V** columns of the SVD: the first column gives the main direction of variation, and the second column the direction of almost constant value. For instance, with the second subcomposition we find

```
> s1 = clr(x[,c("CaO","P2O5","Na2O")])
>  s1c = s1 - mean(s1)
> v = svd(s1c)$v[,2]
> var( scalar(s1, v) )

[1] 0.009853
```

which represents a tiny 0.6 % of the total variation. Such projections may uncover interesting relationships in the form of equilibrium equations. Following with this example, we have to look for a rescaling of the vector v that provides nice, round coefficients:

```
> names(v)=names(s1)
> v/min(abs(v))

   CaO   P2O5   Na2O
 2.269 -1.269 -1.000

> 4*v/min(abs(v))

   CaO   P2O5   Na2O
 9.076 -5.076 -4.000
```

The procedure applied is to divide the vector by the smallest (in absolute value) of its components and then look for a nice rounding, a quite tricky procedure. In this case, we may finally write that

$$C^t \simeq \frac{(\mathrm{CaO})^9}{(\mathrm{P_2O_5})^5 \cdot (\mathrm{Na_2O})^4} \approx \left(\frac{(\mathrm{CaO})^2}{\mathrm{P_2O_5} \cdot \mathrm{Na_2O}} \right)^{4.5} \tag{6.2}$$

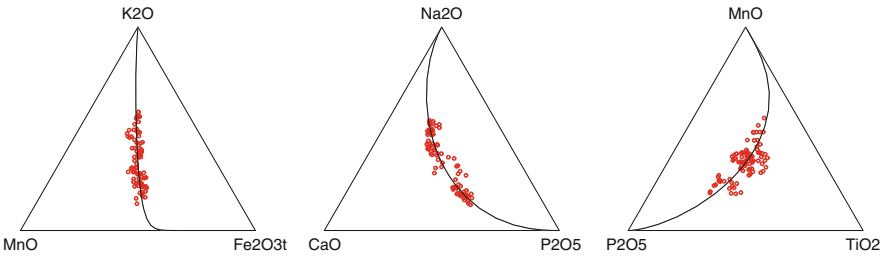


Fig. 6.2 Three ternary diagrams where the biplot (Fig. 6.1) suggested a possible one-dimensional pattern. This pattern is shown as a *line*.

is almost constant. Note that the second expression, the most simplified one, is just balancing Ca against P and Na. Similar relationships for the other selected subcompositions may be derived following the same procedure. This is left as exercise for the reader.

Finally, the links associated to the first and the last of these subcompositions are quite orthogonal; thus we can expect a certain independence of these two patterns. To check it, let us compute the first **U** singular vector on each of the selected subcompositions, with the centered, clr-transformed data:

```
> s1 = clr(x[,c("MnO","Fe2O3t","K2O")])
>  s1 = s1 - mean(s1)
> s2 = clr(x[,c("P2O5","TiO2","MnO")])
>  s2 = s2 - mean(s2)
> u1 = svd(s1)$u[,1]
> u2 = svd(s2)$u[,1]
> cor(u1,u2)
[1] -0.04206
```

The resulting correlation is a very small value, clearly almost zero.

In summary, with a proper exploration of the biplot, we uncovered several relations of low variance (proportional components, or quasi-constants of equilibrium) in several subcompositions. Note that if we hope to find these relations by looking at all possible ternary diagrams, we would have to explore 720 plots (the different combinations of 10 elements taken 3 at a time), which is quite a boring task. Thus, the biplot is a very useful tool in the exploration of the codependence structure of a compositional dataset.

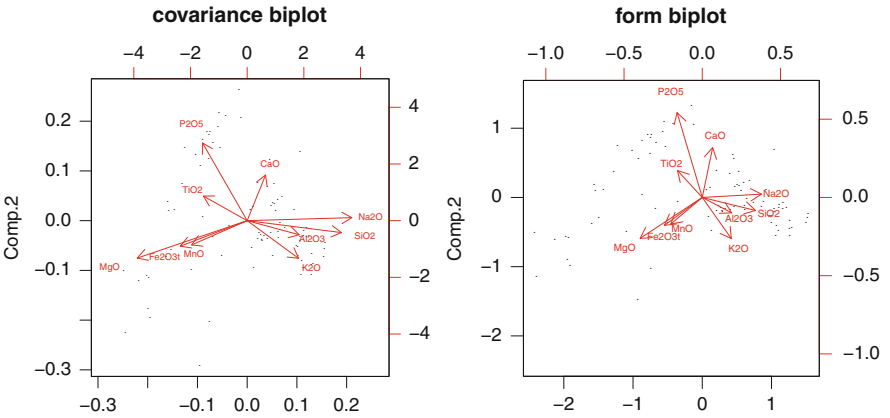


Fig. 6.3 Biplot variants (*left*, covariance biplot; *right*, form biplot) of a principal component analysis of the glacial sediment dataset. The *right-hand* biplot shows a graphical representation of the communalities

6.1.3 The Scree Plot and the Form Biplot

When interpreting a biplot, either compositional or classical, it is very important to remember that the quality of the representation is controlled by the proportion of preserved variability (6.1). Two graphical results allow to visualize this proportion and which variables are better captured. These are the *scree plot* and the so-called *form biplot*, obtained for $\alpha = 0$. The scree plot is a bar plot (or a line plot) of the variance of the principal components, i.e., of the numbers d_{ii}^2: a good biplot is associated with a scree plot where the first two heights are very large when compared with the remaining bars.

In a form biplot, the proportion of variability of a variable is given by the length of its ray: thus a perfectly represented variable has a length of one unit, and a badly represented variable has a very short ray. Figures 6.3 and 6.4 show examples of covariance and form biplots and a scree plot. This proportion of explained variability for each component is called *communality*.

Example 6.2 (Plotting a compositional PCA). Calling `princomp`, giving as argument an "`acomp`" dataset, produces a compositional PCA, stored in an "`princom.acomp`" object. This is a fork of the **R** basic "`princomp`" object. Several plotting functions are available for these two object classes. A direct plot, like

```
> plot(pcx)
```

generates the scree plot (Fig. 6.4), a representation of the squared diagonal elements of **D**. Similar representations are provided by

```
> plot(pcx, type="screeplot")
> plot(pcx, type="variance")
```

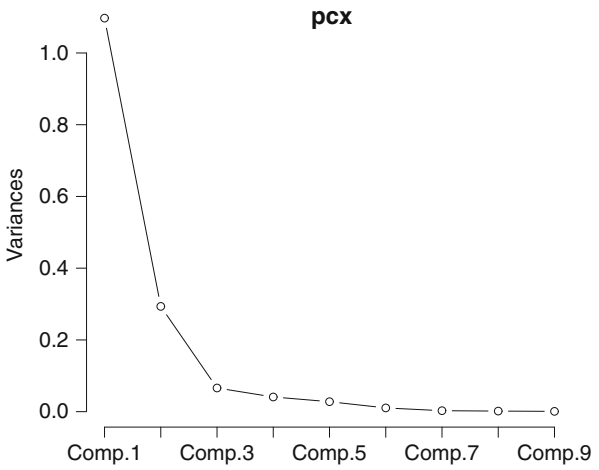


Fig. 6.4 PCA scree plot of major oxide geochemical composition of glacial sediments

The first option generates a line plot, and the second one a bar plot. The sum of the first two variances against the total variance is the proportion of preserved variation:

```
> sum(pcx$sdev[1:2]^2)/sum(pcx$sdev^2)
```

```
[1] 0.9026
```

in this case quite a good value. The covariance biplot of the data is obtained with either

```
> biplot(pcx)
> plot(pcx, type="biplot")
> plot(pcx, type="biplot", scale=1)
```

The optional argument `scale` controls whether we get a covariance biplot ($\alpha = 1$), useful to assess the codependence between variables, or else a form biplot ($\alpha = 0$) where the ray length of each variable represents its communality:

```
> dots = rep(".",times=nrow(x))
> biplot(pcx, xlabs=dots, scale=0)
```

These biplots are displayed in Fig. 6.3. Note how Na_2O and CaO have the same length in the form biplot (thus, we can equally trust or distrust any conclusions involving one or the other), while in the covariance biplot, the ray of Na_2O is around twice as long as the one of CaO (thus, the clr variance of the former is around 2 times the clr variance of the latter).

6.1.4 Loadings and Evolution Plots

A PCA generates more results than those represented on a biplot or a scree plot. But to understand and interpret them, it is good to recast the SVD in terms of the internal operations of the simplex, perturbation, and powering. In this way, the ith

composition (the ith row of our compositional dataset) can be written as the center plus some deviations along the directions of the principal components:

$$\mathbf{x}_i = \mathbf{m} \oplus \bigoplus_{j=1}^{D-1} (u_{ij} \cdot d_j) \odot \mathbf{w}_j,$$

where now each vector $\mathbf{w}_j = \text{clr}^{-1}(\mathbf{v}_j)$ is the inverse clr of the jth column of $\mathbf{V}$, the matrix of principal component loadings. Thus, the vectors of loadings can (and should) be interpreted as compositions, so conveying only information about increments or decrements of a part with respect to another. This means that we cannot interpret the loadings isolated, but at least in pairs, as illustrated in the following example.

The clr loadings (the matrix $\mathbf{V}$) may be accessed by using the function `loadings` on a "`princomp`" object, whereas their compositional counterparts (the vectors $\mathbf{w}_j$) are stored in its "`Loadings`" element. But as the signs of the principal components are arbitrary, we can also access the inverse composition $\mathbf{w}_j^{-1} = \text{clr}^{-1}(-\mathbf{v}_j)$ in the "`DownLoadings`" element. Both "`Loadings`" and "`DownLoadings`" are only available in "`princomp.acomp`" objects, not in the basic **R** "`princomp`" results.

Three different plots can be obtained with PC loadings:

- *Barplots of* $\mathbf{w}_j$: these are stacked bar chart representations of each PC loadings expressed as compositions, i.e., as positive sectors summing up to one. These must be compared with the neutral composition $\mathbb{1}$, having all components equal to $1/D$. Thus, the larger the contrast between the ith components of $\mathbf{w}_j$ and $1/D$ is, the stronger influence the jth PC exerts over the ith variable *relative to the remaining variables*. This may be quite difficult to interpret; therefore, we introduce the following alternative representation.
- *Barplots of ratio loadings*: these are sideways bar charts of each pairwise ratio of two components, where each PC is represented as a bar, showing its multiplicative contribution onto that particular ratio. If the ith bar for the ratio A/B is larger (smaller) than one, we may say that the ratio A/B increases (decreases) along the ith PC, and the larger (smaller) this bar is, the faster this increase (decrease) is.
- *Evolution plots* are scatterplots of the scores $(u_{i1} \cdot d_1)$ of the first PC (on the X-axis) against the observed compositions $\mathbf{x}_i$ and/or the predicted composition $\mathbf{m} \oplus (u_{i1} \cdot d_1) \odot \mathbf{w}_1$ (on the Y-axis), each component with a different color/symbol. Further PCs can be also represented, but then one must use the order k residual and predicted compositions, respectively:

$$\mathbf{e}_i = \mathbf{x}_i \ominus \mathbf{m} \ominus \bigoplus_{j=1}^{k-1} (u_{ij} \cdot d_j) \odot \mathbf{w}_j$$

$$\hat{\mathbf{x}}_k = \mathbf{m} \oplus (u_{i1} \cdot d_1) \odot \mathbf{w}_k$$

against the scores of the kth PC. Note that, as usual, these quantities can be computed with classical sum and products on the clr- or ilr-transformed dataset. Evolution diagrams typically show exponential decay/growth patterns.

For a correct interpretation of these patterns, the compositions must be presented as relative to a reference: e.g., assuming that a given component will not change. Then, we can say that growing curves correspond to components that increase faster (or decrease slower) than the reference component.

Example 6.3 (Visualizing and interpreting compositional PCA loadings). Following with the preceding example, let us interpret the first two principal components (PCs). The first following line returns their clr coefficients (which sum is checked to be zero in the second line):

```
> loadings(pcx)[,1:2]
```

```
          Comp.1   Comp.2
SiO2     0.42551 -0.10401
TiO2    -0.19735  0.21449
Al2O3    0.23464 -0.12300
MnO     -0.25085 -0.21190
MgO     -0.49627 -0.32678
CaO      0.08298  0.39815
Na2O     0.47260  0.02635
K2O      0.23196 -0.32847
P2O5    -0.20126  0.67795
Fe2O3t -0.30196 -0.22278
```

```
> colSums( loadings(pcx) )
```

```
    Comp.1     Comp.2     Comp.3     Comp.4     Comp.5
-5.135e-16  7.876e-16 -4.365e-15  3.421e-15  3.809e-15
    Comp.6     Comp.7     Comp.8     Comp.9
 6.696e-15  3.192e-16 -8.056e-14 -7.511e-14
```

These coefficients are not free to vary in a ten-dimensional space, but linked to each other through this null sum. We cannot thus interpret a low value as an absence of dependence! For instance, in this example, we should not think that the first PC is not affecting or depending on CaO, in spite of having a very low loading: we can nevertheless say that the Na_2O vs. CaO log ratio increases along the first PC with a slope of $0.473 - 0.083 = 0.39$. Or we may also say that this first PC is almost unable to change the ratio P_2O_5/TiO_2 (because their loadings difference is almost zero), whereas along the second PC, this log ratio increases with a slope of 0.463. A graphical representation of these slopes can be obtained with `plot(..., type="relative")` and `plot(..., type="loadings")` (Fig. 6.5): here the first line just leaves more space for the axis labels. The remaining lines generate the four diagrams.

These are not available as a command, but can be quickly constructed with `matplot` and `abline`. The following code generates one of these figures, for the first PC:

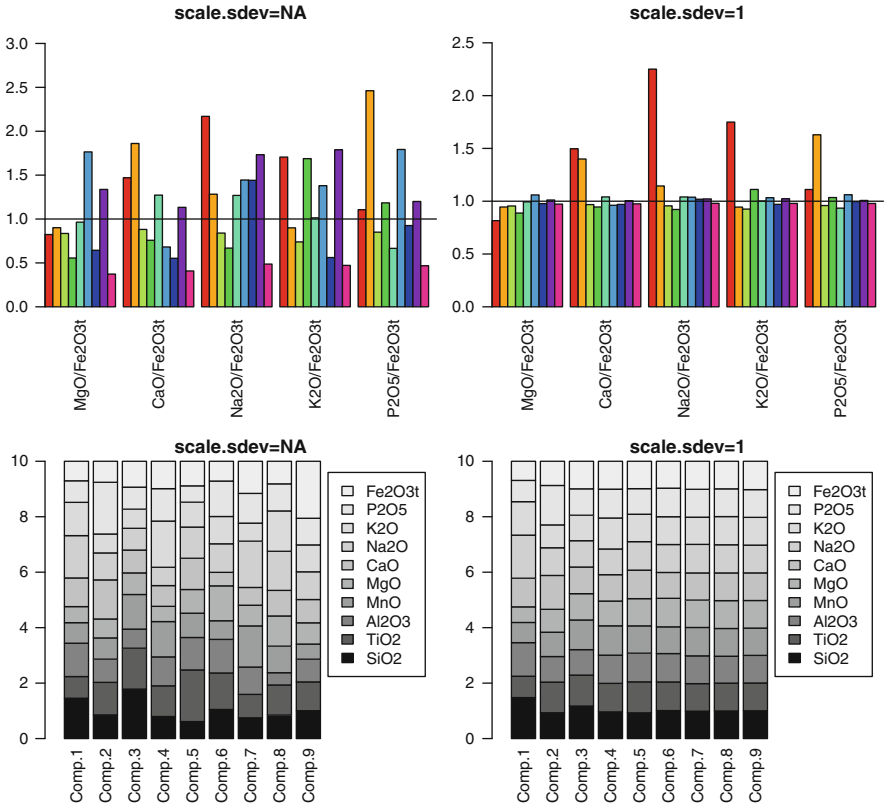


Fig. 6.5 Bar plots of loadings of a compositional principal component analysis: (*left*) without any scaling (sdev.scaling=NA) and (*right*) with one-unit scaling (sdev.scaling=1). Only the last third part of the actual plot is displayed. For an object `pc` containing the results of a PCA, the *upper row* uses side bars generated with the code `plot(pc, type="relative")`, and the *lower row* contains stacked bars generated with `plot(pc, type="loadings")`. The *left-hand* side panels show the loadings scaled by the corresponding singular value, showing that the bars of higher-order PCs tend to be more similar to 1; i.e., the last PCs are less able to modify the composition than the first PCs. The *right-hand* side panels show all loadings as unit compositional vectors, which allows to see the structure even for the principal components with smaller variation. *Upper plots* (`type="relative"` option) represent the multiplicative factor that each PC exerts over each pairwise ratio of the composition. If a bar has a unitary height for a given PC, that PC is unable to modify the ratio of the two parts involved; otherwise, if the bar is significantly lower (higher) than 1, that PC reduces (increases) the ratio. *Lower plots* (option `type="loadings"`) show the loadings as compositional vectors

```
> comprel2d = function(data, fixedvar){
+   diag(1/data[,fixedvar]) %*% unclass(data)
+ }
> comprel1d = function(data, fixedvar){
+   unclass(data)/data[fixedvar]
```

```
+ }
> fk = pcx$scores[,1]
> vd = pcx$Loadings[1,]*pcx$sdev[1]
>   vd = comprel1d(vd, "Al2O3")
> mn = pcx$Center
>   mn = comprel1d(mn, "Al2O3")
> matplot(fk, log(comprel2d(x, "Al2O3")), pch=19, col=rainbow(10))
> for(i in 1:10){
+   abline(a=log(mn[i]), b=log(vd[i]), col=rainbow(10)[i], lwd=2)
+ }
```

First, we define two convenience functions to recompute a composition relative to one part. Then, we extract the first PC scores and loadings (scaled by its singular value). Finally, we represent all the variables in the same plot, the observations as dots and the evolution model as lines. Alternatively, we may want the diagrams in raw scale, which does not allow us to display the model as a family of lines:

```
> fkdens = seq(from=min(fk)*1.1, to=max(fk)*1.1, length.out=200)
> compdens = clrInv(outer(fkdens, clr(vd))) + pcx$Center
> compdens = comprel2d(compdens,"Al2O3")
> etqy = compdens[length(fkdens),]
> par(mfrow=c(1,2), mar=c(3,3,1,1))
> for(logscale in c("","y")){
+ matplot(fk, comprel2d(x,"Al2O3"),
       pch=19, col=rainbow(10), log=logscale, cex=0.75)
+ matlines(fkdens, compdens, lty=1, col=rainbow(10))
+ }
> text(x=fkdens[length(fkdens)], y=etqy,
                              labels=colnames(x), pos=2)
```

In this chunk, we define an equally spaced sequence within the observed score range of the first PC, estimate the expected composition for each of these scores, and recast it relative to Al_2O_3. Then, we plot the observed data and the expected compositions computed before, in both cases relative to alumina. Finally, we use `text` to place a label at the right end of the plot to identify each curve with its variable. The plot (Fig. 6.6) is once produced in raw scale and once in log scale, which would be equivalent to the result of the preceding chunk.

The visual fit of this line to the data will mostly depend on the proportion of variability captured by the principal component chosen: in this case the first PC captures a 71.2 % of the total variance (Fig. 6.4); thus, the diagram shows a very reasonable fit, and the model can be trusted to understand what is happening within the data. So, keeping Al_2O_3 stable in the direction of increase (reduction) of SiO_2, we must also increase (reduce) Na_2O while keeping Na_2O almost constant and reducing (increasing) all other components that mostly discards that weathering exerts a significant control over these sediments geochemistry (at least related to the first PC), as such a process would yield a faster decrease of CaO than the decrease

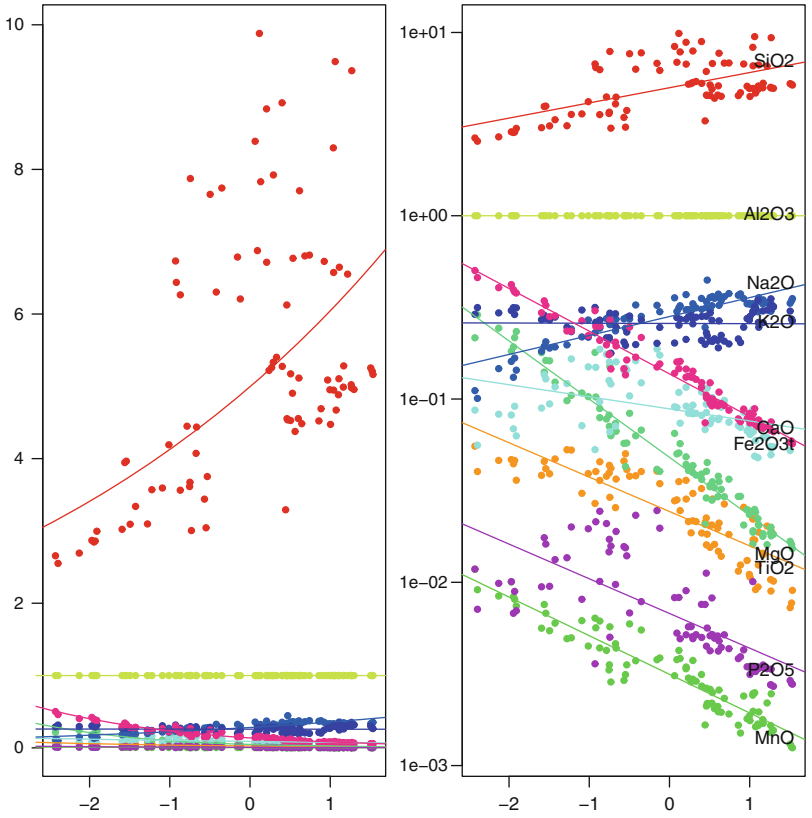


Fig. 6.6 Evolution plots of the composition of glacial sediments along the first principal component, assuming a conservative Al_2O_3: (*left*) in raw scale, (*right*) in log scale

of K_2O in the direction that SiO_2 decreases. This is due to the faster alteration of plagioclase with respect to K-feldspar.

6.2 Cluster Analysis: Detecting Natural Groups

Most techniques described in this book should be applied to homogeneous, single-population datasets (or multiple-population data already splitted in subpopulations). Unfortunately, natural datasets are seldom homogeneous: often there are several subgroups, corresponding to different, unknown subpopulations with a distinct behavior. The existence of such groups may be hinted in several descriptive plots (Figs. 4.9 and 6.1 are clear examples). *Cluster analysis* (Fahrmeir and Hamerle, 1984; Krzanowski, 1988; Mardia et al., 1979) is a technique devised to uncover them

and to classify the data in the several groups found. In this book we will present only hierarchical cluster analysis.

Two steps are needed to generate a cluster analysis, and each one has several variants; thus we must choose a "method" of computation for each step:

1. *Obtain a distance matrix between the individuals to compare* using function `dist(X, method)`, where `X` is the matrix containing (*by rows*) the individuals to compare and `method` is a string with the name of the desired distance to use.
2. *Apply an agglomeration technique*, a hierarchical algorithm which first considers all samples as isolated groups, then it mixes the two nearest groups in a joint group, and the process is repeated upwards until all observations are pooled into one single group. Here the useful command is `hclust(d, method)`, where `d` is the output of the preceding step (a distance matrix between individuals), and `method` is a string with the name of the desired method of computation of the *distance between groups*.

Details on these steps and their methods are given in the next two subsections.

6.2.1 Compositional Distances

Most distance measures for multivariate datasets can be meaningfully generalized to compositions if they are applied to the clr-transformed dataset. This is automatically done, when the function is applied to an "`acomp`" object. The following values can be thus given as `method` argument and obtain a sensible distance:

"`euclidean`" (the default) the classical distance measured as the square root of the sum of the squared differences between the coordinates of the two individuals being compared. With compositional data, this happens to be equivalent to the Aitchison (1986) distance (2.5), as mentioned in Note 2.2).

"`minkowski`" a generalization of the Euclidean distance:

$$d(\mathbf{x},\mathbf{y}) = \sqrt[p]{\sum_{i=1}^{D} |\mathrm{clr}_i(\mathbf{x}) - \mathrm{clr}_i(\mathbf{y})|^p},$$

where the root/power is not necessarily $p = 2$. Unlike the Euclidean distance, the Minkowski distance depends on the chosen basis of the vector space. Thus, generally speaking it is only meaningful in compositions when based on the clr-transformed dataset, as this log-ratio transformation is the only one which is permutation invariant (Sect. 2.2). For the same reasons, the next two distances should only be applied in general terms to the clr transformation.

"`manhattan`" obtained as the Minkowski distance with $p = 1$: this is just the sum of the projections along the axes, following the trajectory needed while driving along a city block grid (other names of this distance are *taxicab* and *city block*, all very visual).

"maximum" obtained as the Minkowski distance with $p \to +\infty$: it amounts to taking only the largest of the projections along the axis.

These last distances may also be applied to ilr-transformed data in the case that the ilr transformation used has a clear interpretation. Another important distance is the *Mahalanobis distance*; this is the Euclidean distance in a dataset linearly transformed to having uncorrelated, unit-variance variables. This distance is not implemented as a method of `dist`, but it can be obtained for compositions with the composed command `dist(svd(ilr(x))$u)`, computing the Mahalanobis distance of the ilr transformation. Theoretically, the Mahalanobis distance of any ilr and alr transforms is equivalent. A Mahalanobis distance of the clr transformation is not defined due to the singularity of the covariance matrix, and computing it is numerically unstable. However, it must be equivalent to the preceding ones and may be computed with a black-box ilr.

6.2.2 Agglomeration Techniques

As agglomeration methods of `hclust`, we might select one of the following (among other):

"complete" ("average" or "single"): the distance between two groups is the maximum (average or minimum) distance between the individuals of the two groups.

"ward" : the distance between two groups is the distance between the averages of each group, weighted by the harmonic average of the number of points in each group.

As the compositional nature of the dataset has been already taken into account, the agglomeration criteria work exactly as with conventional data. The reader is thus referred to the help obtained with `? hclust` to decide which `method` is more adequate for its problem.

Example 6.4 (Visualizing clusters). The whole sequence of a cluster analysis may be, for instance,

```
> xc = acomp(x)
> dd = dist(xc)
> hc = hclust(dd, method="ward")
> plot(hc)
```

where the last step generates a graphical output (a dendrogram) showing the hierarchical agglomeration structure. The decision on which is the level to cut a dendrogram (and thus define the number of existing groups) is quite subjective, though a good idea is to look for long height intervals without agglomeration and cut the tree inside them. An interactive cut can be obtained with the code

```
> h = locator(1)$y
> rect.hclust(hc, h=h)
> gr = cutree(hc, h=h)
> plot(x,col=gr)
```

The first line waits until the user clicks somewhere in the graphic and reads the height coordinate there. Then the two following lines draw rectangles around the groups and generate the grouping variable (`gr`), for further use. The last line would display the grouping in a ternary diagram.

6.2.3 Compositional Q-Mode Cluster Analysis

Cluster analysis can also be used to cluster *variables* instead of individuals. In this case we must pass to `hclust(...)` a "distance" or measure of association between variables, such that perfect association is represented as zero and perfect independence as $+\infty$. For compositional data, such a measure has already been introduced: the variation matrix (4.3). The only preliminary step to be taken is the internal conversion of the variation matrix into a proper distance matrix recognizable by **R**, with the command `as.dist`. Classical approaches to Q-mode clustering straightforwardly applied to compositions, like using the arc-cosine of the correlation matrix of the clr-transformed data, should be avoided, as the obtained distance between any two variables would not be subcompositionally coherent.

Example 6.5 (Clustering compositional variables). Following with Example 6.2, we want to cluster the variables, to detect similar patterns of variation between them. As distance we use the variation array, and as clustering criterion the Ward method:

```
> dd = as.dist(variation(x))
> hc = hclust(dd, method="ward")
> plot(hc)
```

The result (Fig. 6.7) suggests to consider three groups of parts:

- The felsic-mineral components (Na, K, Al, Si), already identified as a quasi-constant subcomposition
- The mafic-mineral components (Mg, Mn, Fe) plus Ti, also seen quite proportional in the biplot analysis
- Ca and P, main constituents of apatite, an accessory but relatively typical mineral of granites

Note that in Example 6.3, the first principal component is roughly speaking a balance between the first two groups of parts here defined, while the last group identifies the second principal component. It seems thus that the geochemical

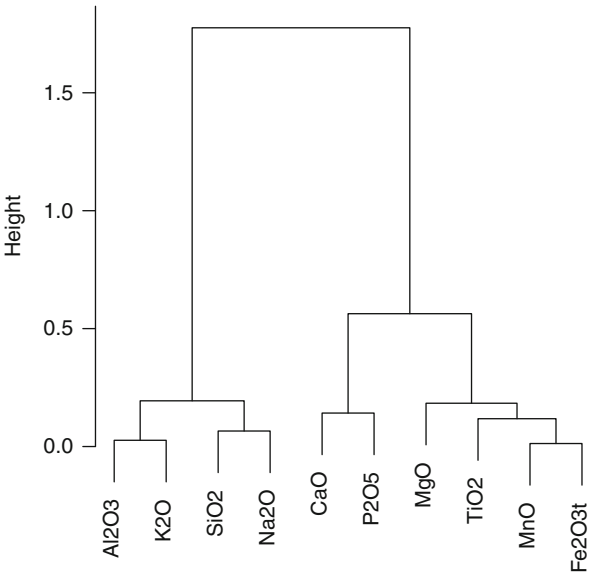


Fig. 6.7 Dendrogram of association of variables of the dataset of geochemical composition of glacial sediments. This is obtained with Ward algorithm using the variation array as distance matrix between parts

variability of these sediments is strongly controlled by the abundance of several minerals. This hypothesis cannot be further checked, because we do not have the available mineralogical composition of these sediments.

6.3 Discriminant Analysis

Let us assume that our dataset is split in *known* subpopulations; i.e., that complementarily to the (compositional) data, we have a grouping variable. Discriminant techniques aim at finding the best way of separating the groups using functions of the available variables. Like regression (Chap. 5), linear discriminant analysis (LDA) uses only linear functions of the data (i.e., projections), whereas quadratic discriminant analysis (QDA) can also use pairwise products of all available variables (i.e., quadratic forms). Multinomial logistic regression, though being also a discriminant technique, will be briefly treated in the next section.

The principle of working on coordinates allows us to apply *any* existing statistical method to the dataset of orthonormal coordinates of our observations and back-transform the results accordingly. Discriminant analysis is not an exception; thus we will easily obtain an LDA or QDA giving an ilr-transformed dataset to the functions computing these discriminant analyses, provided by the package "`MASS`".

Example 6.6 (A linear discrimination of river waters). We consider for this example the dataset about the hydrochemical composition of some water samples from the Llobregat river basin (NE Spain Otero et al., 2005), included in the package

"compositions". Due to the variability of its landscape and land uses, this basin was split in four subbasins: the upper Llobregat course (dominated by Mesozoic calcareous landscape), the Cardener river (as the preceding one but also influenced by intensive potash mining leachings), the Anoia river (dominated by detrital and gypsum-rich tertiary sediments), and the lower Llobregat course (a mixture of all the other). For these characteristics, every hydrochemist would expect the Piper plot to offer a satisfactory separation of the four basins. This is a joint representation of the main cations (Na^++K^+, Ca^{2+}, Mg^{2+}) and anions (Cl^-, HCO_3^-, $SO_4^=$):

```
> data(Hydrochem)
> xc = acomp(Hydrochem, c(7:10,14,17,18))
> river = Hydrochem$River
> mean(xc)
```

```
     Na       K      Mg      Ca      Cl     SO4    HCO3
0.10686 0.01203 0.03800 0.14119 0.15875 0.22932 0.31385
attr(,"class")
[1] acomp
```

```
> table(river)
```

```
river
         Anoia       Cardener LowerLLobregat UpperLLobregat
           143             95            135            112
```

The preceding lines first load the dataset from the package, then extract the compositional information and the classification factor, and show us a minimal summary of each. Let us now try how well the four basins can be separated with an LDA, from package "MASS":

```
> library(MASS)
> res = lda( x=data.frame(ilr(xc)), grouping=river  )
> res
```

```
Call:
lda(data.frame(ilr(xc)), grouping = river)

Prior probabilities of groups:
         Anoia       Cardener LowerLLobregat UpperLLobregat
        0.2948         0.1959         0.2784         0.2309

Group means:
                    V1      V2     V3    V4     V5    V6
Anoia          -1.865  0.3002 1.1991 1.023 1.5805 1.077
Cardener       -1.319 -0.1931 0.9341 1.120 0.7930 1.124
LowerLLobregat -1.201 -0.4504 0.6791 1.209 0.6661 1.030
UpperLLobregat -1.742  0.5297 1.9259 0.666 1.5362 1.939
```

```
Coefficients of linear discriminants:
       LD1     LD2     LD3
V1 -1.2979  0.9541  0.1805
V2  0.3070  1.1034 -1.6925
V3 -1.6289 -2.1734 -3.2735
V4 -0.4393  0.2806  0.9020
V5  2.6479  1.0893  3.1112
V6 -0.3161 -0.7964  3.7893

Proportion of trace:
   LD1    LD2    LD3
0.6329 0.3449 0.0222
```

Function `lda` only requires data given in a data frame, so the ilr-transformed composition must be converted accordingly with command `data.frame`. The result of this chunk, `res`, is an "`lda`" object with several interesting elements. By default, in the `lda` summary, some of these results (group means and the coefficients of the discriminant functions) are reported with respect to the six black-box ilr coordinates used in the computation. That pretty much obscures the interpretation of results. We need a recasting of the ilr numbers into compositions or equivalent clr coefficients:

```
> ilrInv(res$means, orig=xc)
```

```
               Na      K         Mg       Ca     Cl      SO4
Anoia          0.10709 0.007665 0.04138 0.1293 0.14366 0.3250
Cardener       0.12437 0.019267 0.03864 0.1330 0.20722 0.1814
LowerLLobregat 0.14269 0.026102 0.03515 0.1112 0.23873 0.1679
UpperLLobregat 0.05512 0.004694 0.03077 0.1845 0.07331 0.2174
               HCO3
Anoia          0.2459
Cardener       0.2961
LowerLLobregat 0.2782
UpperLLobregat 0.4341
attr(,"class")
[1] acomp
```

```
>   V = ilrBase(xc)
>   rownames(V)= colnames(xc)
> t(ilr2clr(t(res$scaling), V=V))
```

```
         LD1     LD2     LD3
Na    0.9262 -0.6365  0.1538
K    -0.9093  0.7129  0.4092
Mg    0.3844  1.3896 -1.7914
Ca   -1.7471 -2.0210 -4.1893
Cl   -0.8276  0.1750 -0.3459
SO4   2.4660  1.1173  2.2554
HCO3 -0.2926 -0.7373  3.5082
```

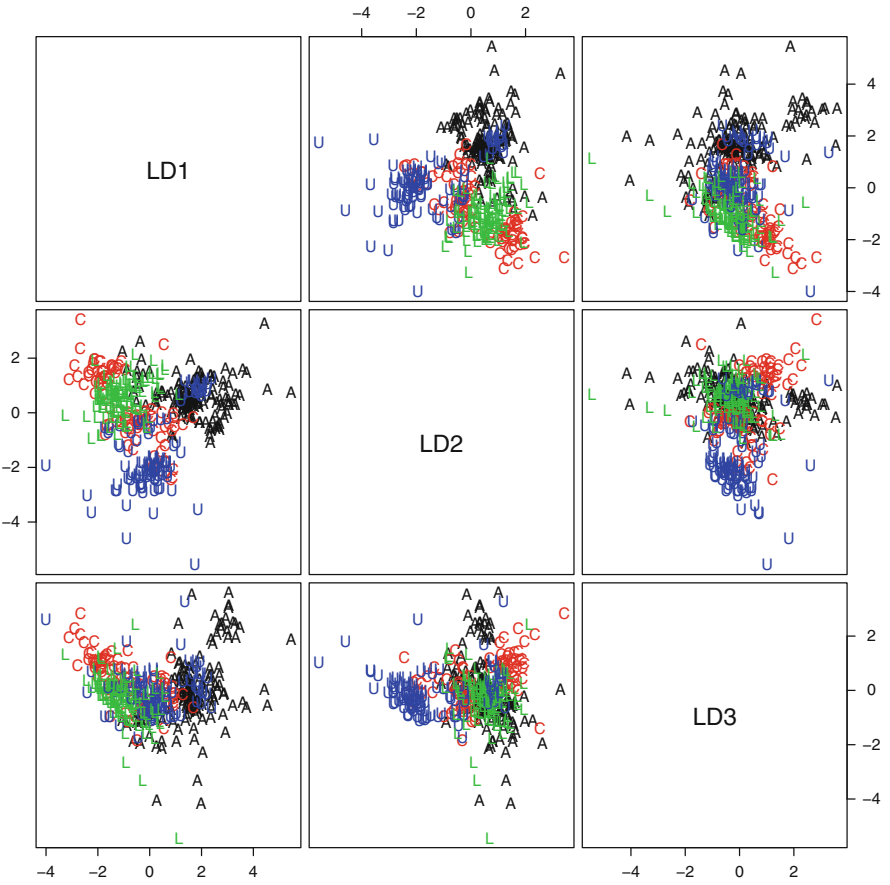


Fig. 6.8 Scatterplot of the linear discriminant functions for the hydrochemical dataset. Each LD function is computed as a projection onto one direction of the simplex. See text for details

In this way we see that, on average, the upper Llobregat river is much richer in HCO_3^- or Ca^{2+} than the other, as high $SO_4^=$ percentages characterize the Anoia river and high Cl^- or K^+ indicate a Cardener or lower Llobregat sample, where "high/low" is always relative to this 7-part subcomposition. To complement the interpretation of the coefficients, we can also plot the `lda` object, with a little bit of preprocessing and formatting to get a clearer plot:

```
> grint = as.integer(river)
> pairs(res, abbr=1, col=(1:4)[grint], cex=1.2)
```

Figure 6.8 tells us that the first LDfunction reasonably splits the Anoia river from the remaining subbasins, and we see that this first function is strongly determined by the

$SO_4^=$ to Ca^{2+} relation, the main components of gypsum. In the same way, the second LDfunction, mostly a $Mg^{2+}{\cdot}K^+$ to Ca^{2+} ratio, characterizes the upper Llobregat basin waters (as being relatively Ca-rich). It is also easy to see that the lower Llobregat course is placed in the middle of the plot, as expected for something which is a mixture of the remaining data. Finally, it is worth noting that the Cardener basin is not homogeneous, as one can clearly identify two subgroups: one quite similar to upper Llobregat or Anoia samples and another extremely rich in those elements with negative coefficient in the first LDfunction and positive in the second LDfunction (as this group occupies the upper left part of the diagram). These elements are K^+ and Cl^-, especially in contrast against Na^+ (which presents the opposite pattern of signs). One can also notice that the boundaries between groups would be more adequately described as curves, thus we should think on using QDA.

Apart of using the principle of working on coordinates, we can also derive the LDA and QDA fundamental equations directly for compositions, using the concepts of projection and quadratic form within the simplex and the normal distribution on the simplex introduced in Chap. 3. Assume a partition of a compositional dataset Z on K groups. For each group, $\mathbf{Z}_i$ represents the subset of data within group i, with a size of N_i; also, $\mathbf{m}_i$ and $\boldsymbol{\Sigma}_i$ are the true mean and covariance structure of each group, and $\bar{\mathbf{z}}_i$ and $\mathbf{S}_i$ denote their empirical counterparts, computed from $\mathbf{Z}_i$.

For LDA, we assume that each group has normal distribution on the simplex, with a different mean, but all groups share the same *pooled* variance, estimated as

$$\mathbf{S}_P = \frac{\sum_{k=1}^{K} N_k \mathbf{S}_k}{\sum_{k=1}^{K} N_k}.$$

For QDA, each group has its own mean and variance. In both cases, we may write the likelihood that a new composition $\mathbf{z}_*$ belongs to each group as

$$L(\mathbf{z}_* \in i) \propto \Pr[\mathbf{z}_* | \mathbf{m}_i, \boldsymbol{\Sigma}_i] \propto \frac{1}{\sqrt{|\boldsymbol{\Sigma}_i|}} \exp\left[-\frac{1}{2}\mathrm{ilr}(\mathbf{z}_* \ominus \mathbf{m}_i) \cdot \boldsymbol{\Sigma}_i^{-1} \cdot \mathrm{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_i)\right],$$

where the $\boldsymbol{\Sigma}_i$ is expressed in the same orthonormal basis used in the ilr transformation. In LDA, these variance matrices are all equal and estimated with $\mathbf{S}_P$, whereas in QDA each one is estimated with the empirical variance within the group $\mathbf{S}_i$. Computing each of the K likelihoods, we obtain a vector, which thanks to Bayes' theorem can be perturbed by some prior probabilities of group membership, p_i^0. This provides the final posterior probabilities that $\mathbf{z}_*$ belongs to each group:

$$\Pr[\mathbf{z}_* \in i] = \frac{(p_i^0/\sqrt{|\boldsymbol{\Sigma}_i|}) \exp\left[-(1/2)\mathrm{ilr}(\mathbf{z}_* \ominus \mathbf{m}_i) \cdot \boldsymbol{\Sigma}_i^{-1} \cdot \mathrm{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_i)\right]}{\sum_{j=1}^{K} (p_j^0/\sqrt{|\boldsymbol{\Sigma}_j|}) \exp\left[-(1/2)\mathrm{ilr}(\mathbf{z}_* \ominus \mathbf{m}_i) \cdot \boldsymbol{\Sigma}_i^{-1} \cdot \mathrm{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_i)\right]},$$

In other words, considering that probability vectors are themselves compositions and obey the same rules described in Chap. 2, we may see the log-posterior prior probabilities as a sort of clr coefficients:

$$l_i(\mathbf{z}_*) = \ln(p_i^0) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| - \frac{1}{2}\text{ilr}(\mathbf{z}_* \ominus \mathbf{m}_i) \cdot \boldsymbol{\Sigma}_i^{-1} \cdot \text{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_i).$$

Again, the only things that matter in the resulting vector $\mathbf{l} = [l_i]$ are the differences between two components i and j, i.e., the log-odds of classification of a sample $\mathbf{z}_*$ between groups i and j:

$$\begin{aligned} F_{(i,j)}(\mathbf{z}_*) &= l_i(\mathbf{z}_*) - l_j(\mathbf{z}_*) = \ln\frac{p_i^0}{p_j^0} - \frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}_i|}{|\boldsymbol{\Sigma}_j|} \\ &\quad - \frac{1}{2}\left[\text{ilr}(\mathbf{z}_* \ominus \mathbf{m}_i) \cdot \boldsymbol{\Sigma}_i^{-1} \cdot \text{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_i) - \text{ilr}(\mathbf{z}_* \ominus \mathbf{m}_j) \cdot \boldsymbol{\Sigma}_j^{-1} \cdot \text{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_j)\right] \\ &= a_{ij} + \mathbf{b}_{ij} \cdot \text{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_{ij}) + \text{ilr}(\mathbf{z}_* \ominus \mathbf{m}_{ij}) \cdot \mathbf{C}_{ij} \cdot \text{ilr}^t(\mathbf{z}_* \ominus \mathbf{m}_{ij}) \end{aligned} \tag{6.3}$$

$$\mathbf{m}_{ij} = \frac{1}{2} \odot (\mathbf{m}_i \oplus \mathbf{m}_j)$$

$$a_{ij} = \ln\frac{p_i^0}{p_j^0} - \frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}_i|}{|\boldsymbol{\Sigma}_j|} - \frac{1}{8}\text{ilr}(\mathbf{m}_j \ominus \mathbf{m}_i) \cdot (\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_j^{-1}) \cdot \text{ilr}^t(\mathbf{m}_j \ominus \mathbf{m}_i)$$

$$\mathbf{b}_{ij} = -\frac{1}{2}\text{ilr}(\mathbf{m}_j \ominus \mathbf{m}_i) \cdot (\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1}) \tag{6.4}$$

$$\mathbf{C}_{ij} = -\frac{1}{2}(\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_j^{-1}) \tag{6.5}$$

Function $F_{(i,j)}(\mathbf{z}_*)$ is called the discriminant function of i against j. By taking $F_{(i,j)}(\mathbf{z}_*) = 0$, we can find the set of points defining the boundary between groups i and j. Several cases can be distinguished, according to the properties of matrix $\mathbf{C}_{ij}$:

- If $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_j = \mathbf{S}_p$, the quadratic form $\mathbf{C}_{ij} = \mathbf{0}$ becomes zero and $a_{ij} = \ln(p_i^0/p_j^0)$. In this case, the boundary is just a line orthogonal to the vector joining the centers of both groups, $\mathbf{b}_{ij}$. This is the LDA case.
- If the eigenvalues of $\mathbf{C}_{ij}$ have all the same sign, then the border is an ellipsoid. The internal group to the ellipsoid is controlled by this sign: if it is negative, then group i is the inner group, and vice versa, if the eigenvalues are positive, the inner group is j. In the case of three parts (two dimensions), this ellipsoid can be drawn as an ellipse inside the ternary diagrams.
- If the eigenvalues of $\mathbf{C}_{ij}$ have different signs, then the border is an hyperboloid or a paraboloid if one of the eigenvalues is zero. Again, for three parts we could draw these borders as hyperbolas and parabolas within the simplex.

6.4 Other Multivariate Techniques

There is no place in a book such as this one to treat in detail the myriad of existing statistical and data mining techniques. The examples of adaptation of the most used multivariate techniques to compositional data explained in this chapter and Chap. 5 may guide in other applications. What follows is a brief guide to adapting some of these other methods for compositions, intended for those readers who are already familiar with these techniques.

6.4.1 Canonical Correlation

If one has two sets of variables observed on the same individuals, this technique may be used to explore which correlations exist between the two families of variables. Roughly speaking, it is an interpretation of the SVD of the correlation matrix between the two random vectors associated to each variable set. If one of these two sets is compositional, we should apply the same principles as in PCA and work thus with a clr-/ilr-transformed set. Computationally, one may be forced to use ilr to avoid the singularity problems of clr, but for interpretation and display (e.g., in the form of a biplot), results should be reexpressed in clr or as compositions. Note that both sets may be compositional, which calls for using two clr/ilr transformations. In **R**, *canonical correlation* is available in `cancor` . This illustration example explores the relationship between major oxide composition and trace element composition of the glacial sediments:

```
> xmaj = acomp( GeoChemSed[,4:13] )
> itr = c("Ba","Cr","Ga","Nb","Nd","Pb","Rb","Sr","Y","Zn","Zr")
> xtr = acomp( GeoChemSed[,itr] )
> Xmaj = ilr(xmaj)
> Xtr = ilr(xtr)
> res = cancor(data.frame(Xmaj),data.frame(Xtr))
> res$xcoef = t(ilr2clr(t(res$xcoef), x=xmaj))
> res$ycoef = t(ilr2clr(t(res$ycoef), x=xtr))
> res$xcenter = ilrInv(res$xcenter, orig=xmaj)
> res$ycenter = ilrInv(res$ycenter, orig=xtr)
> biplot(x=res$xcoef, y=res$ycoef)
```

The last four lines would recast the canonical variables from ilr to clr coefficients and the variable means to compositional means. We could finally use a biplot to graphically represent both sets of coefficients.

6.4.2 Logistic Regression

As an alternative to LDA/QDA, one may consider that an individual D-part composition $\mathbf{x}$ may be classified in a K-class problem according to a multinomial realization with probability vector $\mathbf{p} = f(\mathbf{x})$. Being a vector of relative positive

quantities adding up to one, $\mathbf{p}$ is also a composition from the K-part simplex. This prediction function $f(\cdot)$ is chosen as an affine linear function from $\mathbb{S}^D$ onto $\mathbb{S}^K$, which coefficients may be estimated by maximum likelihood. Logistic regression implementations already acknowledge the compositional nature of $\mathbf{p}$ by predicting the log-odds of the several classes against a reference one (in **R**, the first class), i.e., by predicting $\text{alr}(\mathbf{p})$. Thus, we just add another log-ratio transformation on the side of the predictor, $\text{ilr}(\mathbf{x})$, and use the classical machinery between these two sets of log-ratio-transformed quantities. In **R**, *logistic regression* is available in `multinom` from package "`nnet`". The next chunk builds upon Example 6.6:

```
> library("nnet")
> auxcomp = scale(ilr(xc), center=TRUE, scale=FALSE)
> dat = data.frame(river, auxcomp)
> frm = river~ V1+V2+V3+V4+V5+V6
> res = multinom(formula=frm, data=dat)
> coef(res)
> ilr2clr(coef(res)[,-1], x=xc)
```

We have centered (but not scaled) the data first, for an easier interpretation of the coefficients. Note that Anoia row is missing: that river is used as the reference category. We may thus see that the log-odds of having a Cardener sample against an Anoia sample increase with higher Ca or Cl proportions, or lower Na and SO4 proportions (and vice versa). The intercept may be interpreted as the log-odds of each river against Anoia at the center of the dataset: these are giving the highest positive value to Cardener; thus, this is the most likely river at the center of the dataset.

6.4.3 Geostatistics

Geostatistics is a family of techniques devised to treat data with spatial dependence. The observations are modeled as regionalized variables $Z(x)$, where an observation could in principle be taken at any location x. However, this regionalized variable is typically only observed at a finite number of locations $x_1, \ldots, x_n$. A typical final aim is to interpolate the data to unobserved locations x and to provide a measure of uncertainty for this interpolation Matheron (1965). The spatial dependence is first analyzed and modeled by second-order moment descriptions, like covariance functions or variograms. The *variogram* $2\gamma(h)$ is the variance of the difference of the values $Z(x)$ and $Z(y)$ at two locations x, y separated by a distance h; i.e., for $h = \|x - y\|$, we have

$$2\gamma(h) = \text{var}(Z(x) - Z(y))$$

It is typically assumed that:

- The expected value is constant and does not depend on the location.
- The variance of the difference only depends on the distance h and not on the actual locations.

This concept is called *intrinsic stationarity* and ensures that the variogram is well defined. For a more detailed treatment of geostatistics, see, e.g., Bivand et al. (2008) (a book from this UseR series), Cressie (1991) and Chilès and Delfiner (1999) (for the mathematical background), or Wackernagel (1998) (about multivariate geostatistics). Tolosana-Delgado et al. (2011) present an adaptation of these principles, concepts, and tools to work with compositional data. This section is but a summary of that work. The variogram can be estimated from the data by averaging the squared differences of pairs of observations in different distance classes:

$$\hat{\gamma}(h) = \frac{1}{N(h)} \sum_{(i,j)\in N(h)} (Z(x_i) - Z(x_j))^2$$

where $N(h)$ denotes the set of pairs of observations with distance approximately h:

$$N(h) = \{(i, j) : \|x_i - x_j\| \approx h\}$$

The precise definition of "being approximately h" is implementation dependent.

Then a model for valid variograms has to be fitted to that empirical variogram function $\hat{\gamma}(h)$. Since variogram fitting is difficult to automatize, the fit must be typically inspected visually afterwards. Once a satisfactory variogram model is obtained, it can be used for interpolation and interpolation error computation by ordinary kriging. *Ordinary kriging* is a method obtaining the best linear unbiased interpolation for an unsampled location based on a known variogram. Results are displayed in maps and can be used for further decisions. Here "best linear unbiased" means smallest mean square error among all those unbiased estimators built as linear combinations of the available data. Nonlinear kriging techniques exist but are typically considered very advanced methodology and are outside the scope of this section.

How do we apply this concept to compositions? Several strategies of variogram modeling have been proposed:

- A simple strategy could be to take coordinates of the composition and to apply classical geostatistical techniques to each coordinate separately. This is a simple but suboptimal approach. The results are, e.g., dependent on the chosen basis and cannot exploit codependence in the composition.
- Use an alr (or ilr) transform and model the multivariate variograms by spectral techniques (Pawlowsky-Glahn and Olea, 2004). Spectral multivariate variogram fitting however is a quite complicated methodology. Alternatively if we do not want to use spectral techniques, we may try multivariate modeling techniques

on any set of coordinates. That could, e.g., be achieved with standard non-compositional tools from the "gstat" package (Pebesma, 2004) in **R**. See Bivand et al. (2008) for details.

- Finding inspiration on the variation matrix, the approach implemented in "compositions" is to model the set of variograms for all possible pairwise balances by a joint model (Tolosana-Delgado et al., 2009). The advantage of this technique is its simple interpretation and usage. The disadvantage is that there is only a limited implementation in "compositions" and nowhere else.

To illustrate this approach we use the jura dataset from the "gstat" package Pebesma (2004), which has been extensively studied in a non-compositional ("rplus") way by Goovaerts (1997):

```
> require(gstat)
> data(jura)
> colnames(prediction.dat)

 [1] "Xloc"    "Yloc"    "Landuse" "Rock"    "Cd"      "Co"
 [7] "Cr"      "Cu"      "Ni"      "Pb"      "Zn"

> colnames(validation.dat)

 [1] "Xloc"    "Yloc"    "Landuse" "Rock"    "Cd"      "Co"
 [7] "Cr"      "Cu"      "Ni"      "Pb"      "Zn"

> colnames(juragrid.dat)

[1] "Xloc"    "Yloc"    "Landuse" "Rock"
```

The jura dataset consists of three datasets: prediction and validation subsets, together with a dense grid, where the geology is given. For simplicity we will not use the geological information, use both datasets together, and restrict ourselves to a ternary subcomposition:

```
> X <- cbind(c(prediction.dat$Xloc,validation.dat$Xloc),
+            c(prediction.dat$Yloc,validation.dat$Yloc))
> Z <- acomp(rbind(prediction.dat,validation.dat)
                                    [,c("Cd","Cu","Ni")])
```

The set of the variograms of pairwise balances can be generated through the `logratioVariogram` command:

```
> lrv <- logratioVariogram(Z,X,maxdist=1.0,nbins=12)
```

The `maxdist` argument controls the largest distance in which the variogram will be estimated. The `nbins` argument controls the number of distance classes. These arguments have reasonable defaults, but are chosen here to highlight the structure in these variograms. To display the variograms one can use an appropriate `plot` method:

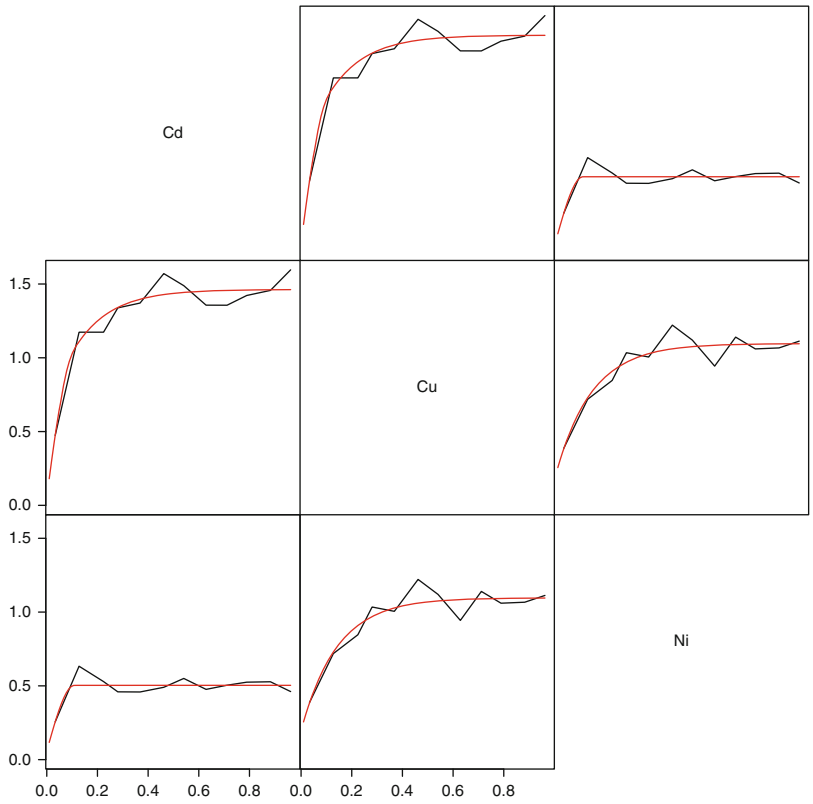


Fig. 6.9 Pairwise balance variograms of the jura dataset. The *zigzag lines* are the estimated empirical variograms. The *smooth lines* are the fitted model

```
> opar <- par(cex=1.25)
> plot(lrv)
> par(opar)
```

This results in a diagram similar to Fig. 6.9 but without the fitted model: The following step is to define a model to fit this variogram. The compositions package implements a quite flexible model of linear coregionalization, which is parameterized through a formula:

```
> lrvModel <- CompLinModCoReg(~nugget()
                              +R1*sph(0.15)+R1*exp(0.1),Z)
```

The composition `Z` must be given, in order to extract from it its dimension and column labels. Possible formula terms are:

- `nugget()`: introducing a general nugget effect. A nugget effect models a discontinuity at very short distances.
- `exp(r)`: models an exponential decay of dependence (an exponential variogram).

- `sph(r)`: models a dependence with limited range (a spherical variogram).
- `gauss(r)`: models a very continuous dependence structure with a sigmoidal decay (a Gaussian variogram).
- `lin(r)`: models a Brownian motion type of decay of dependence (a linear variogram). The range is an estimate of the distance to reach a variance of 1.
- `pow(r)`: models are of the form h^r, generalizing the Brownian motion type of dependence. Here $0 < r < 2$ (generalized linear variogram or power variogram).

In any case `r` has to be replaced by an initial guess of a range of influence for this substructure of the variogram. The precise formulas can be found in the package help for the functions (`vgram.nugget`, `vgram.exp`,...). Each of the components is implicitly multiplied with an arbitrary variance matrix of the appropriate size. This variance matrix can be restricted to a rank 1 matrix by multiplying the variogram component with the symbol `R1` (for "rank 1") in the formula.

In our case a look at the empirical variogram shows two different ranges, but no multiple structures in the individual variograms. The space of a 3-part composition is two dimensional. These facts suggest introducing two components with different ranges, but limiting each one to rank 1. The shape of the longer range seems an exponential variogram. The shorter range structure suggests rather a limited-influence (spherical) structure. We thus choose a model consisting of these two different structures. Obviously other models could fit similarly well.

Function `vgmFit2lrv` provides an automatic fitting procedure for variogram model based on the `nlm` nonlinear minimization routine in basic **R**. It tries to minimize a squared logarithmic discrepancy between the model and the empirical variogram:

```
> vgmModel <- vgmFit2lrv(lrv, lrvModel, print.level=0)
> vgmModel

$nlm
$nlm$minimum
[1] 0.2324

$nlm$estimate
[1]  0.1605 -0.2503 -0.1169  0.1086 -0.5248 -0.2420  0.3302
[8] -0.6560  0.3932

$nlm$gradient
[1]  4.544e-07  2.474e-07 -1.465e-07  1.267e-07 -1.308e-07
[6]  8.153e-07 -2.814e-07 -2.433e-07 -4.627e-07

$nlm$code
[1] 1

$nlm$iterations
[1] 89
```

```
$vg
function (h, sPSD1 = c(0.160454927040757, -0.250288321003883,
-0.116891404703496), rsph2 = 0.10857277169615,
sR2 = c(-0.524764507705152, -0.241960364506509),
rexp3 = 0.33023788094404, sR3 = c(-0.655960605455565,
0.393177110888582))
vgram.nugget(h = h) %o% parametricPosdefClrMat(sPSD1) +
                                        vgram.sph(h = h,
    range = rsph2) %o% parametricRank1ClrMat(sR2) +
                                        vgram.exp(h = h,
    range = rexp3) %o% parametricRank1ClrMat(sR3)
```

The result combines the output of the `nlm` fitting procedure (for control purposes, it is better to leave `print.level=1`) with a function in the `vg` element, which is the actual fitted variogram model. The default arguments of this latter function provide the fitted model parameters. To compute the actual meaning of the parameters, one must use the transformation functions `parametricPosdefClrMat` or `parametricRank1ClrMat`, respectively. These functions allow to encode matrices in unbounded vector arguments through their Cholesky decomposition. Note that the result is specified in the set of default ilr coordinates.

The model can be plotted along with the empirical variogram by adding it as an extra argument to the `plot` method for "`logratioVariogram`" objects:

```
> plot(lrv,lrvg=vgram2lrvgram(vgmModel$vg)) #$
```

The `vgram2lrvgram` translates a multivariate variogram model into a model for the variograms of pairwise balances. Having a look at the result of this command (Fig. 6.9), we see that the model provides a reasonable fit. Thus, we will use it in subsequent kriging steps.

Finally, an interpolation will be computed for every point of the juragrid dataset. We thus store the grid coordinates into a matrix:

```
> Xnew <- cbind(juragrid.dat$Xloc,juragrid.dat$Yloc)
```

The actual interpolation is performed by multivariate *ordinary kriging* through the `compOKriging` command taking as arguments:

- The observed compositions (`Z`)
- The observation locations (`X`)
- The interpolations location (`Xnew`)
- The chosen variogram model `vg`

```
> KP <- compOKriging(Z,X,Xnew,vg=vgmModel$vg) #$
> names(KP)

[1] "X"   "Z"   "err"
```

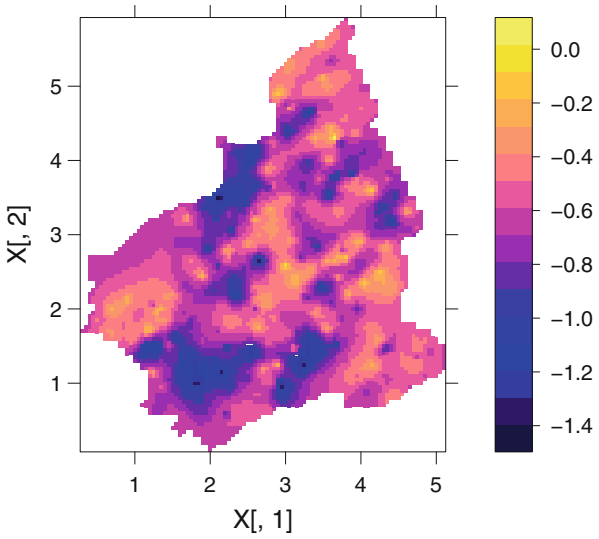


Fig. 6.10 Map of interpolated composition represented by the $(Ni\ Cd)/Pb$ balance

In the result, `X` contains the new observation locations given as `Xnew`. `Z` provides the actual interpolations. The *kriging variance*, i.e., the expected squared difference of interpolation and true value, can be additionally obtained by giving the argument `err=TRUE` to `compOKriging`. It is then given in the `err` part of the output.

These results can, e.g., be presented in maps for individual balances. In this case the nicest map is given by the $(Ni\ Cd)/Pb$ balance (Fig. 6.10):

```
> require(lattice)
> print(levelplot(balance(Z,~(Ni*Cd)/Pb)~X[,1]+X[,2],
+     KP,col.regions=bpy.colors(20)))
```

The function `print` is used here only to force the graphic to appear. Graphics from the "`lattice`" package are not stored in a file otherwise. Apart of displaying balances, individual components could be plotted if needed, with

```
> levelplot(Z[,"Ni"]~X[,1]+X[,2],KP,col.regions=bpy.colors(20))
```

However, be aware that, according to the compositional principles, this does not make much sense. In particular the results are closed to 1, while the original data were in ppm.

References

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. London: Chapman & Hall (Reprinted in 2003 with additional material by The Blackburn Press), 416 pp.

Bivand, R. S., Pebesma, E. J., & Gomez-Rubio, V. (2008). *Applied spatial data analysis with R*. New York: Springer.

Chilès, J. -P., & Delfiner, P. (1999). *Geostatistics—modeling spatial uncertainty*. Series in probability and statistics. New York: Wiley, 695 pp.

Cressie, N. (1991). *Statistics for spatial data*. New York: Wiley, 900 pp.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika, 1*, 211–218.

Fahrmeir, L., & Hamerle, A. (Eds.). (1984). *Multivariate statistische verfahren*. Berlin: Walter de Gruyter, 796 pp.

Gabriel, K. R. (1971). The biplot—graphic display of matrices with application to principal component analysis. *Biometrika, 58*(3), 453–467.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Applied geostatistics series. New York: Oxford University Press, 483 pp.

Krzanowski, W. J. (1988). *Principles of multivariate analysis: A user's perspective*. Oxford statistical science series (Vol. 3). Oxford: Clarendon Press, 563 pp.

Mardia, K. V., Kent, J. T., & Bibby, J. M. (1979). *Multivariate analysis*. London: Academic, 518 pp.

Matheron, G. (1965). *Les variables régionalisées et leur estimation—une application de la théorie des fonctions aléatoires aux sciences de la nature*. Paris: Masson et Cie., 305 pp.

Otero, N., Tolosana-Delgado, R., Soler, A., Pawlowsky-Glahn, V., & Canals, A. (2005). Relative vs. absolute statistical analysis of compositions: A comparative study of surface waters of a Mediterranean river. *Water Research, 39*(7), 1404–1414.

Pawlowsky-Glahn, V., & Olea, R. A. (2004). *Geostatistical analysis of compositional data*. London: Oxford University Press.

Pebesma, E. J. (2004). Multivariable geostatistics in S: The gstat package. *Computers & Geosciences, 30*, 683–691.

Tolosana-Delgado, R., van den Boogaart, K., & Pawlowsky-Glahn, V. (2009). Estimating and modeling variograms of compositional data with occasional missing variables in R. In: *StatGIS'09, Geoinformatics for environmental surveillance Workshop*, June 17–19, Milos (Greece), proceedings. https://www.stat.aau.at/Tagungen/statgis/2009/StatGIS2009_Tolosana%20Delgado_2.pdf.

Tolosana-Delgado, R., van den Boogaart, K., & Pawlowsky-Glahn, V. (2011). Geostatistics for compositions. In: V. Pawlowsky-Glahn, A. Buccianti (Eds), *Compositional data analysis: theory and applications*, Chapter 6 (pp. 73–86). Chichester: Wiley.

Wackernagel, H. (1998). *Multivariate geostatistics, an introduction with applications* (2nd ed.). Berlin: Springer, 291 pp.

Chapter 7
Zeroes, Missings, and Outliers

Abstract The presence of irregular data, i.e., zeroes, missings, and outliers, has consequences for plotting, descriptive statistics, estimation of parameters, and statistical tests. Since all of these methods depend on every value of the dataset, we cannot ignore them in any task. Some ad hoc procedures are needed, with the same aims as the classical methods, but which can still be computed despite the existence of irregular data. It is necessary to *augment* the basic concepts (e.g., the concept of expected value) to give them a meaning when there are irregular values. The current state of the art of the treatment of irregularities in compositional data analysis is far from being a closed subject and can improve a lot in the near future. The package only provides a set of tools limited to detect, represent, and briefly analyze such irregular values, either missing values, zeroes, or outliers. This chapter provides nevertheless additional background material to enable the reader to carefully treat datasets with irregular data.

7.1 Descriptive Analysis with and of Missings

7.1.1 Types of Zeroes and Missing Values

Missing values are appearing quite often in compositional data and are often (and erroneously) coded with a zero value. Most commonly, missings occur when one of the variables could not be observed, e.g., because the actual amount was below the detection limit of the measurement procedure. However, there are many other kinds of reasons why a datum can be missing, and their treatment largely depends on these reasons. Thus, several types of missing values must be defined, all with their own inner logic and corresponding statistical consequences. The following classification is implemented in the package:

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7_7, © Springer-Verlag Berlin Heidelberg 2013

- *Missing (completely) at random (MAR)*
  A *missing at random* occurs when the value was not observed for reasons (stochastically) independent of its actual value and genetic process. This is the least problematic form of missing values, although it still implies certain challenges for estimation and graphical representation. A subclass of MAR is the *missing completely at random* (*MCAR*), which is further assuming that the missingness is also independent of other variables that might be simultaneously observed. "`compositions`" codes a MAR as the constant `MARvalue=NaN`. There is no particular MCAR encoding in the package.
- *Structural zero (SZ)*
  We say that a given variable has a *structural zero* in a given observation when that part is not properly defined or simply cannot exist due to some deterministic reason. For instance, this may happen due to chemical incompatibility with other present components or because that component was artificially removed during the sampling. From a practical point of view, a structural zero can often be treated in a similar way to a MAR, though the interpretation will be totally different. "`compositions`" codes a SZ as `SZvalue=-Inf`.
- *Below detection limit (BDL)*
  For small amounts, the measurement method might not be precise enough to actually distinguish the true value from zero. In this case, most labs report the value as *below detection limit*. In the presence of values below the detection limit, serious problems arise in compositional data analysis. The current methodology is to replace the lost value by some fraction of the detection limit itself. This is far from providing convincing results, because of the arbitrariness of the replacement and because reclosure with replaced values may change all portions of the composition.
  A below-detection-limit value can be coded in two ways in "`compositions`". If the detection limit (i.e., the lowest value that actually would be reported) is known, a BDL value is coded as the negative of the detection limit value. In this way, the values can be afterwards adjusted if the composition is rescaled or reclosed into a subcomposition. If the detection limit is unknown, the BDL is encoded by a 0.
- *True zero*
  A *true zero* occurs when a given part is not present, even though it was considered in the measurement. It typically only shows up in count compositions, when the affected variable has a low probability of occurrence and the total number of counts is also relatively low. In this sense, true zeroes resemble BDLs, because we can expect that the lost part could be finally observed by increasing the total number of counts. True zeros are extremely rare in other compositions and cannot be treated in the standard framework of "`acomp`" compositions. They can be handled in the framework of "`rcomp`" and "`ccomp`" and are then coded with 0, just like a BDL with unknown detection limit.
- *Missing not at random (MNAR)*
  A missing value is a *missing not at random* (*MNAR*) if the chances to have missed it depend (stochastically or deterministically) on the actual value missed. For instance, if the measurement procedure can break down when subject to too high

stimuli, it will tend to fail more often when measuring high values; thus a missing will more easily occur when the true value is high. In a given sense, BDLs can also be taken as MNAR cases: the values are deterministically non-reported if they are below a threshold. General MNAR is the most difficult type of missings to model, because we need to know why and how they occur, i.e., to know the probability law of missing a value given that value. In "`compositions`", a missing not at random is encoded as `MNARvalue=NA`.

Missing values can be stored encoded in datasets of types `acomp`, `rcomp`, `ccomp`, `aplus`, and `rplus`. Encoded missings in the datasets are displayed using the acronyms (MAR, SZ, BDL, MNAR) or by <*DetectionLimit*:

```
> data(SimulatedAmounts)
> x <- acomp(sa.missings)
> acomp(x[1:10,])
```

```
      Cu      Zn      Pb
 [1,]     MAR 0.43383 0.5662
 [2,]    MNAR 0.05261 0.9474
 [3,] 0.02365 0.22930 0.7471
 [4,] 0.15712 0.33356 0.5093
 [5,] 0.07898     BDL 0.9210
 [6,] 0.15533 0.47056 0.3741
 [7,] 0.10273 0.08076 0.8165
 [8,]    MNAR 0.05342 0.9466
 [9,] 0.19485 0.36820 0.4369
[10,]    MNAR    MNAR    MAR
attr(,"class")
[1] acomp
```

```
> acomp(c(-0.001,-Inf,Inf,0.5,0,NA,NaN))
```

```
[1] <0.002     SZ    ERR  1.000    BDL   MNAR    MAR
attr(,"class")
[1] acomp
```

7.1.2 Ternary Diagrams with Missing Values

Plotting a dataset with missings tries to encode the missing information as informative as possible. This is illustrated in Fig. 7.1 , displaying a simulated dataset with all types of missing values presented in Sect. 7.1.1:

```
> opar<-par(mar=c(3,3,0,0))
> data(SimulatedAmounts)
> x <- acomp(sa.missings)
> plot(x)
> par(opar)
```

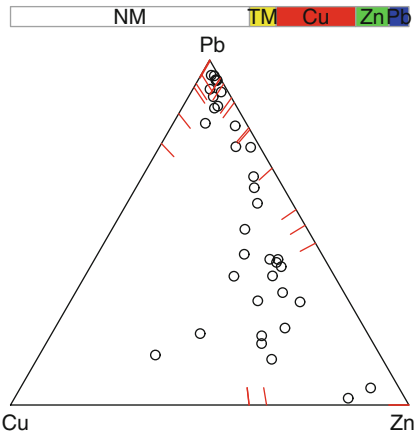


Fig. 7.1 Example of a ternary diagram with missing values

The resulting ternary diagram uses two mechanisms to display missing values. In an upper subpanel, called the *missings plot*, it displays how many observations present missing values in all combinations of variables possible, by a horizontal bar plot: `NM` stands for "No Missings" (i.e., how many observations do we have where all variables were available); TM for `Totally Missing`, which means that at least two of the three values are missing such that no subcompositional information can be computed; and the names of the variables stand for those cases, where only that component is missing.

If only one of the three components is missing, we still know the ratio of the other two, which corresponds with a straight line passing through the vertex of the lost variable: the true value of that observation is an unknown point on that line. To avoid overplotting, this line is replaced by a tick mark at the axis joining the two observed variables. This is called a *missings tick mark* and is controlled by the `missingTck=` parameter.

The missing bar plot and the missings tick marks can be suppressed by giving the optional parameter `plotMissings=FALSE`.

7.1.3 Representing Missing Values Themselves

The `missingSummary` command summarizes the number of missings in the various variables into a table:

```
> missingSummary(x)

         missingType
variable NMV BDT MAR MNAR SZ Err
      Cu  44   0   6    7  3   0
      Zn  53   1   4    1  1   0
      Pb  54   0   5    0  1   0
```

Plotting a `missingSummary` object shows a bar plot displaying the relative amount of missings in each of the variables:

```
> plot(missingSummary(x), xlim=c(0,ncol(x)+2))
```

From a more exploratory perspective, a compositional dataset can be converted to a dataset of factors classifying the type of missingness on each variable and for each case, by using the command `missingType`:

```
> xMissType = missingType(x)
> data.frame(x, xMissType)[1:5,]
```

```
       Cu      Zn     Pb Cu.1 Zn.1 Pb.1
1     NaN 0.43383 0.5662  MAR  NMV  NMV
2      NA 0.05261 0.9474 MNAR  NMV  NMV
3 0.02365 0.22930 0.7471  NMV  NMV  NMV
4 0.15712 0.33356 0.5093  NMV  NMV  NMV
5 0.07898 0.00000 0.9210  NMV  BDT  NMV
```

This may help in getting a clearer idea of which variables concentrate the irregularities and which should be a reasonable treatment of them.

It could happen that some missing values occur simultaneously in two or more variables, or somehow associated in a complex way (e.g., a Cu under the detection limit could be preferentially associated with missing Zn at random or any other combination we might think of). To visually display these patterns of association of missing type across variables, we can first apply the missing type classifier to the dataset, then extract the joint probability table of missings:

```
> missingCombinations <- table(as.data.frame(xMissType))
```

and finally display it with any graphical tool for categorical variables, like a `mosaicplot`:

```
> mosaicplot(missingCombinations +0.01, shade=TRUE)
```

Note that the extra argument `shade=TRUE` produces the coloring of boxes according to positive association of the combination of missings (blue colors, solid borders) against negative associations (red colors, dashed borders). The addition of 0.01 to the whole table is an esthetic trick to get clearer mosaic tiles; the reader should remove it, once the plot is understood.

7.1.4 Programming with Missings

On a more advanced level, the package also includes a set of functions to allow experienced **R** users to program their own irregular data-handling routines. Command `hasMissings` checks whether there are any missing values in a dataset.

The vectorized predicates `is.BDL`, `is.MAR`, `is.MNAR`, `is.SZ`, `is.NMV` (standing for *n*on-*m*issing *v*alues), `is.WZERO` (*w*ider *zero*, including BDL, SZ, and true zeroes), and `is.WMNAR` (*w*ider *MNAR*, including both BDL and MNAR values) can be used to identify the different types of missing in programming. To create and manipulate the observed subcomposition projectors, one can use the commands `missingProjector` and `sumMissingProjector`. Finally, if one needs an artificial dataset with missing values for testing purposes, the command `simulateMissings` returns the dataset given as argument, but with some missings of the specified types.

7.2 Working with Missing Values

7.2.1 Imputation

Since many methods do not work with missings, the first approach to missings is to remove the problem by replacing them with *informed guesses*. This approach is called *imputation*. A good guess has to be based on a model of why the values are missing and what the conditional distribution of these values given the observation might be. All these are dependent on the type of missingness. Also imputations typically fall in a circular reasoning: the imputation depends on some parameters (e.g., mean and variance of the theoretical distribution), but in order to estimate them, we should have already the imputed values.

Several methods for imputation are available. These methods range from simple imputation rules, like replacing the missing component with the arithmetic or geometric mean of its non-missing observations (Aitchison, 1986; Martín-Fernández et al., 2003) or by a given fraction (e.g., 2/3) of the detection limit, to more elaborated computations of conditional expectations (Palarea-Albaladejo and Martìn-Fernández, 2008), like realizations of conditional distributions according to linear models extracted from the non-missing components and additional covariables.

We may distinguish at least four different levels of imputations:

1. Imputation doing not too much harm, e.g., replacing zeros with a fixed fraction of the detection limit or MAR values by the average of the observed values. Replacements for BDL and SZ in this line were already proposed by Aitchison (1986) and further studied by Martín-Fernández et al. (2000) and Fry et al. (2000).
2. Imputation keeping expectation, e.g., replacing a zero with a fraction of the detection limit computed from a model of the underlying distribution. For instance, if one assumes that the tail of the distribution between the detection limit and zero resembles a triangular distribution, then values below the detection limit should be replaced by 2/3 of this detection limit.
3. Imputations keeping expectations and variances, e.g., replacing a zero with a random outcome of a conditional distribution of the unobserved value

computed from an estimated model of the underlying distribution. This has been proposed using an estimation-maximization algorithm (Palarea-Albaladejo and Martìn-Fernández, 2008).

4. Imputation keeping *conditional* expectations and variances, e.g., by including regressors into the conditional distribution model of the preceding case. Tjelmeland and Lund (2003) presented one of these cases, where BDLs were treated with Markov-Chain-Monte-Carlo (MCMC) methods within a geostatistical model of spatial dependence.

In any case, it is important to accurately estimate the parameters of interest from the incomplete dataset, before any imputation can be done. Imputation is thus more a way of hiding missings in the interpretation than really a way to solve their problems.

In all cases, the missing part is replaced by a new nonzero number, breaking the closure relationship of all components adding to 1 or 100 %. The dataset is subsequently reclosed by dividing every component by the new joint sum. This is called *multiplicative imputation*, because every observed component is multiplied with a reclosing factor. In this way, the ratios of the unaffected components are retained. The *additive* alternative method of subtracting something from the non-missing components would destroy the ratio relationship, which was the only real information in a composition with missing components. If there are further observations below detection limit, the detection limits of those have to be scaled accordingly (Martín-Fernández et al., 2003).

The components are thus not simply concentrations with a given detection limit. The numerical value of the detection limit depends on the closing constant for the given composition. The detection limit is thus typically different for each value in a compositional dataset and changes for each reclosing of the composition, when another value is imputed or a subcomposition is taken. Detection limits are not meaningful as portions but are meant on actually observed concentrations in the original measurement. In each rescaling or closure operation, the detection limits of each of the components thus have to be scaled along with the data. If compositions are perturbed, the detection limits and standard errors have to be perturbed with them. This implies that a component may have different detection limit at each different row, which must be recomputed in every simplicial operation.

An example might illustrate how natural are indeed these considerations. Take the composition $[A, B, C, D] = [0.85, 0.1, 0.05, -0.05]$, where part D was below the detection limit of 0.05, and the subcomposition $\{A, B, C\}$ has been reclosed accordingly. If we remove the first part, this datum must become $[B, C, D] = [0.5, 0.25, -0.25]$; thus the detection limit apparently changed. This is actually just an effect of the relative nature of compositions, as the ratio of the detection limit to any of the observed parts remains unchanged. Have a look at the last line in Sect. 7.1.1 to see another example.

In a real geometry, the effect of imputation or the choice of the imputation of a missing value seems negligible, since the value is anyway bounded between 0 and the almost zero detection limit. From the Aitchison geometry perspective, however, every imputed value will afterwards be used to compute ratios between an observed

value and the unobserved components. And this ratio can vary up to infinity. Even worse, ratios between two components below their detection limits can take any value, as we do not actually know which is larger, and in which proportion. A seemingly harmless imputed value can thus easily turn into a gross outlier.

7.2.1.1 Random Imputation

Theoretically, the perfect imputation is easy to achieve. One simply calculates the conditional distribution of the missing values given: (1) the observed values and (2) their status of missingness. Then, one obtains a random realization of this conditional distribution and replaces the missing values with it. This can be done when one single variable is missing or also with a vector of missing variables. However, imputation has several theoretical difficulties that need our consideration prior to its application in a specific situation.

The first problem is a mere modeling challenge: obviously, different types of missing data, e.g., values not observed and values below detection limit, have different conditional distributions. Likewise, values below different detection limits have different conditional distributions. It is thus important to identify the correct type of missingness.

A more serious problem corresponds to the fact that computing conditional distributions always depends on knowing the joint distribution. But this conditional distribution is never available: having it (including its true parameters) is actually the ultimate goal of any statistical analysis. Typically, we will use a statistical model for the observations, estimate the parameters, and compute the conditional distribution based on the estimated parameters. But again, if we could know the model and estimate its parameters without using the missing data, we would not need to impute them. And if we use a simpler model, we will get imputation which do not represent the original data.

Consider for instance a situation where virtually no gold is in the samples of one region and substantial gold is present in most samples of a different region. Imputations ignoring the dependence would come up with relatively high imputed values not far from the detection limit as they infer the global conditional distribution of gold mostly from the gold-rich area. On the other hand, a more specific model would have almost no chance to correctly estimate as zero the mean gold in the no gold area and would yield some positive number below the detection limit. Imputation is thus almost always a guess, hopefully not too bad.

The most important effect of imputation is its artificial reduction of uncertainty. Let us imagine a complete dataset (with no missings) and its version with some missing values. Obviously, in the first case, we have more information than in the second case. But imagine we apply regression to the first one and obtain a given measure of global residual uncertainty. Then we impute the missing values in the second dataset and compute the same residual uncertainty measure of regression. We will obtain a similar value in the imputed case than in the complete case, or even a smaller one (as we may be replacing a variable missing value by a constant

replacement), while the complete case carries more information and should thus clearly have the minimal uncertainty. Imputation enhances the information coming from the data and model with fake information coming from our guesses. The user should thus use imputation with great care. Any subsequent analysis unaware of imputation will yield results apparently less uncertain than they should be.

Imputation is nevertheless the best choice in many situations, where it does a great job: just adding a little bit of junk information, we may be able to fit a reasonable model to an otherwise unmanageable dataset. This is typically the case when we have only a small number of missing values. But still then the user has the responsibility to understand the method and decide what might have been the effect of the imputation, of the model used in the imputation, and of the measurement errors behind the imputation.

A formally correct use of imputation would be a complete Bayesian approach. We would specify a prior for the parameters of the model and would compute the full posterior distribution of these parameters, together with the predictive distribution of those values missing. However, in this case, the model and parameter inference would not actually need the imputation. They would actually be a (rather immediate) consequence of the model. That is, the Bayesian approach with this model would solve the estimation problem correctly without the imputation.

The special case of this problem is that any imputation is based on parametric distribution models, typically based on an additive-log-ratio-normality assumption. This often makes imputation sensitive to outliers and will always generate distributional tails according to that model. This means that the tail needed for imputation is inferred from the observations in the center of the data. Imputation is thus a special case of a model-based extrapolation. Being model-based, this extrapolation might be quite stable, but it is still not based on the reality of these tails.

A practical problem for imputation is that, even in cases of simple models with additive-logistic-normal distribution, it is not possible to give analytical solutions for the conditional densities of values below the detection limit. Imputation is thus often done in iterative procedures approximating the reality.

Things become really awkward if we take into account the analytical error (Sect. 7.2.8) in the imputation procedure of an additive-logistic-normal model. We must thus ask ourselves whether we are estimating the *true values* (e.g., with an ALN model, cut above the detection limit) or the *observed ones* (e.g., a mixture of ALN and censored normal models, also cut above the detection limit). The presence of an additive error implies that values observed above the detection limit may be actually below it. And indeed, they are going to occur more frequently than its opposite (actual value BDL, observed value above the detection limit), as ALN models will typically have more probability slightly above the detection limit than slightly below it! This means that imputing with the conditional distribution of the true value would not be in accord with the definition of random imputation: the true values have statistical properties different from that of the actually observed data. But the distribution of the observed data might even be non-compositional, giving some probability to negative values (possible effect of a large additive error). Simple

ALN or normal imputation models do not fully capture this complexity. One must in this case resort to extensive simulation techniques of Bayesian hierarchical models.

7.2.1.2 Deterministic Imputation

A special case of imputation is the *deterministic imputation*: one replaces missing values with a deterministic value, typically based on the detection limit or the global data average. This deterministic value can either be computed by a statistic model (e.g., by something like the conditional expectation of the inferred distribution) or by some rule of thumb, like taking, e.g., $2/3$ of the detection limit. Using this second sort of deterministic imputation will introduce some arbitrariness into the data. The challenge is to predict the implications, keeping in mind that we work in a relative geometry, where multiplicative factors matter. It is not possible to define a general behavior of these deterministic imputation rules: there are examples where such a rule serves the case very well, while it is also easy to construct counterexamples, where the rule introduces severe outliers badly determining the result of the analysis.

The advantage of using the conditional expectation rather than a random imputation is that the result of the analysis is no longer random. If the conditional distribution models were correct, linear statistical methods relying on the expected value would still yield unbiased results with this expected value imputation. This would be the common case of estimating regression coefficients. However, methods relying on variances will find too small results:

$$\begin{aligned}\mathrm{var}(X) &= P(A)\mathrm{var}(X|A) + P(A^c)\mathrm{var}(X|A^c) + \mathrm{var}(E[X|\sigma(A)]) \\ &< P(A)\mathrm{var}(X|A) + P(A^c)\underbrace{\mathrm{var}(E[X|A^c]|A^c)}_{0} + \mathrm{var}(E[X|\sigma(A)])\end{aligned}$$

Such underestimation of the variance due to the deterministic imputation may be critical for subsequent statistical tests (e.g., regression coefficient significance test), as they may become too aggressive.

In conclusion, imputation should not be perceived as a simple tool to get rid of the problem of missing data, but rather as a patch covering a major injury. We should actually resort to critical thinking before taking anything for granted.

7.2.1.3 Current Implementation

Currently, "`compositions`" only supports deterministic imputation (type 1) for BDLs using the function `zeroreplace`:

```
> (x <- acomp(c(-0.1,1,1)))

[1] <0.05  0.50  0.50
attr(,"class")
[1] acomp
```

```
> (xZeroReplaced <- zeroreplace(x))

     [,1]    [,2] [,3]
[1,] 0.03333  0.5  0.5
attr(,"Losts")
                                        ...
Containing  1  lost and replaced values!
attr(,"class")
[1] acomp

> attr(xZeroReplaced,"Losts")

     [,1]  [,2]  [,3]
[1,] TRUE FALSE FALSE
```

This function replaces the BDLs by a fraction (parameter `a`, defaults to a constant 2/3) of the detection limits, either explicitly given by a vector `d` or extracted from the dataset if BDLs are encoded as negative values.

7.2.2 Missingness at Random as Projected Information

For MAR missings, van den Boogaart et al. (2006) developed a geometric strategy to deal with missing values, based on projecting each datum to a particular subspace of the simplex, using the Euclidean space structure of Sect. 2.5. Here we consider that each observation without missing values defines a point on $\mathbb{S}^D$. On the contrary, data with missing components define lines, planes, hyperplanes, etc., with dimension equal to the number of missing values in that datum, i.e., contributing to our knowledge of the population properties with less information than a fully observed point. Then, estimation of parameters like the mean vector or the variance matrix can be built taking that deficit of information into account.

This approach is very useful to *represent* missings in plots: in a ternary diagram or scatterplot of components, we will be able to plot missings according to their nature. A MAR or MNAR will appear as a line passing through all points compatible with the values actually observed for the remaining variables. A BDL will be represented as a segment of this line. Structural and true zeroes are left as zeroes, represented along the axes of the ternary diagram.

To estimate the mean, we can associate to each observation a projector matrix from the whole simplex to a subspace without the missing variable. Each projector ensures that its datum does not contribute in any sense to the determination of the mean of the missing variable but still brings information to the mean of the other variables. To estimate the variance, we proceed in a similar way, but associating a projector to each pair of observations. These projectors ensure that each pair of data do contribute only to the determination of the covariances of those variables observed in both data.

The following example shows this strategy implemented automatically in estimation of the mean by using a simulated example:

```
> (mn = acomp(c(1,2,3)))

[1] 0.1667 0.3333 0.5000
attr(,"class")
[1] acomp

> vr = ilrvar2clr( diag(2) )
> x = rnorm.acomp(50, mn, vr)/2
> xm = simulateMissings(x, MARprob=0.10)
> xb = simulateMissings(x, detectionlimit=0.15)
> mean(xm)

[1] 0.2385 0.3516 0.4099
attr(,"class")
[1] acomp

> mean(xb)

[1] 0.2775 0.3239 0.3986
attr(,"class")
[1] acomp
```

Compare the true mean composition with its estimates. Note that the dataset with MAR values is well behaved, while in the dataset with BDLs, this strategy overestimates the mean of the smallest part. Section 7.1.2 applies the ideas of this section to plotting.

7.2.3 Treating Missing at Random

Missing at random (MAR) means that the missingness (the fact of observing or missing a value) is independent of the affected variable, but not necessarily independent of the remaining variables. It is then possible to estimate the probability of missingness and therefore to find a way to correct the problem. However, because of the closure, a given component cannot be independent of the other components in the classical sense. We thus need to redefine the concept of MAR for compositional data analysis. We may realize that the actual information available is the observed subcompositions. Thus, a compositional MAR is a value which is lost depending only on the value of the observed subcomposition. A possible situation leading to MAR values would be when the actual decision of measuring a further component is based on the observations of a prior set of variables. For instance, we might devise a situation where a primary ore metal is very cheap to measure, while a secondary one is very expensive: in this situation, it would be worth measuring the secondary one only when the content of the primary one was high enough.

It is thus necessary to introduce (and estimate) a model for the missingness probability. A *logistic regression* might be a sensible model, eventually including further covariables. This logistic regression could then predict the missingness probability for each of the observations. The following example shows such a fit with simulated data.

First, we simulate a compositional dataset of four parts (with fixed seed for reproducibility) and generate some MAR. Then we fit the logistic regression, with help of the generalized linear model command `glm`:

```
> require(lmtest)
> datset = data.frame(ilr(simMAR[,-4]),
                          y = factor(is.MAR(simMAR[,4])))
>  colnames(datset)

[1] "V1" "V2" "y"

> lrmMAR =  glm(y ~ V1 + V2 , family=binomial, data=datset)
> waldtest(lrmMAR)

Wald test

Model 1: y ~ V1 + V2
Model 2: y ~ 1
  Res.Df Df    F  Pr(>F)
1     97
2     99 -2 11.8 2.5e-05

> ilr2clr(coef(lrmMAR)[-1])

      1       2       3
-0.9265 -2.7526  3.6791
```

A Wald test (in package "`lmtest`") tells us that the regression coefficients are significant. The reader should also try to plot the diagram of the "observed" subcomposition using colors with the MAR status of the 4th part:

```
> # represent the MAR missings as color in the 3-diagram
> aux = ilrInv(datset[,-3])
>  colnames(aux) = c("A","B","C")
> plot(aux, pch=as.integer(datset$y) ) #$
> straight(mean(ilrInv(datset[,-3])), ilrInv(coef(lrmMAR)[-1]))
```

This last line adds the model regression gradient (Sect. 5.2), showing the direction where the probability of having a MAR increases.

To estimate the mean, we would split the coordinates into balances of the observed parts and a balance of the potentially missing vs. the observed parts. The mean balances of the observed parts can be estimated in the classical way. The estimation of the balance of the potentially unobserved parts would be obtained as weighted averages, by using the inverse of the (inferred) probability of obtaining a missing

value for each row. A similar approach would be possible in regression models with the incomplete composition as dependent variable. First, we extract the missing probability and construct the weights:

```
> ## extract the MAR prob:
> mp = inv.logit(predict(lrmMAR))
> w = 1/(1-mp)
```

Next, we build a balance basis isolating the missing part, the D:

```
> V = balanceBase(simMAR, ~D/(A/(B/C)))
```

and finally, we compute a weighted mean for the first balance:

```
> tkObs = is.NMV(simMAR[,4])
> ILR = ilr(simMAR,V)
>  colnames(ILR)=c("V1","V2","V3")
> meanV1 = weighted.mean(ILR[tkObs,"V1"], w=w[tkObs] )
```

The object `tkObs` is used to filter out the data with missings in the mean computation. That mean could be introduced in the balance mean and then the ilr inverted to obtain an estimate of the compositional mean:

```
> bmean = mean(ILR)
> bmean[1] = meanV1
> ilrInv(bmean,V)

      A       B       C       D
0.08422 0.16914 0.25667 0.48996
attr(,"class")
[1] acomp
```

Compare that with the value of `mn`, the true mean.

The logistic regression model could also be used in a deterministic or a random imputation. We may then infer the conditional distribution of the balance of the unobserved part vs. the observed parts by a weighted linear regression model with the observed subcomposition as independent variable (see Chap. 5). But now the weights should be taken as the odds of missing the values, as provided by the logistic regression model:

```
> ## extract the MAR prob:
> w = mp/(1-mp)
> wrm = lm(V1~V2+V3, ILR, w=w, subset = tkObs )
```

A deterministic imputation value could take the regression prediction:

```
> ILR[!tkObs,1] = predict(wrm,
                  newdata = as.data.frame(ILR[!tkObs,]) )
> impMAR = ilrInv(ILR, V=V)
```

Again the object `tkObs` is used to filter out the data with missings in the regression and to select which rows of the first balance must be replaced by their prediction. The final line would invert the balances to obtain an imputed dataset. The following plot would compare the true simulated values of this balance with their predictions:

```
> plot(ILR[,1], ilr(simDAT,V)[,1], col=is.MAR(simMAR[,4])+1)
```

If we would like a random imputation instead of a deterministic one, we could complement the replacement using the linear model prediction with the variability of the residuals:

```
> ILR[!tkObs,1] = predict(wrm,
                newdata = as.data.frame(ILR[!tkObs,]) ) +
+   rnorm(sum(!tkObs), mean=0, sd = sqrt(var(wrm)) )
> impMARrandom = ilrInv(ILR, V=V)
```

Note that all these procedures will be independent of the ilr basis used because of the affine equivariance of linear models.

A Bayesian approach able to deal with MAR values needs an explicit model for the functional form of the dependence of the probability of missing a value given the observed subcomposition. This missingness sub-model might contain further parameters, eventually needing the specification of hyper-prior distributions. A typical sub-model of this kind could be taken from a logistic regression. If the parameters of the sub-model are assumed a priori independent of the parameters of the global model, we can estimate them independently prior to the rest of the model. The estimate of the global model parameters of the composition itself will then depend on this sub-model. The likelihood for a fully observed datum in such a situation would be the product of the probability of not missing it times the full compositional distribution density. For a partly MAR-missing observation, the likelihood would be the product of the observed marginal density times the probability of missing those parts which are not available. Note that all methods explained in the preceding section about MCAR can be applied to MAR once we know the model of missingness.

The practical difficulty arises if several components might be missing at random depending on the observed subcomposition, because several different subsets of variables could be observed. However, it does not really seem reasonable that the probability of the observation of a component should depend on some other component if it is observed and does not depend on it if it is not observed. Thus, in the presence of MAR values in several variables, it might be better to use simpler models with a missing not at random situation.

To graphically represent an observation with a MAR value, we can find inspiration on the projection approach and represent it with a line, either in scatterplots of log ratios or in ternary diagrams. In this last case, such lines join the vertex of the missing variable with the point on the opposite side given by replacing the lost value with a zero and reclosing the observation. Exceptionally, these lines are always straight from both the "`acomp`" and "`rcomp`" perspectives. This is further explained in Sect. 7.1.2.

7.2.4 Treating Missings Completely at Random

The idea behind *MCAR* (missing completely at random) is that each amount is missed or observed with a given probability, independently of anything else. This implies that the distribution of observed subcomposition, conditioned to what is observed, is the same as the marginal distribution of the subcomposition. We can thus understand a MCAR simply as less information. Every situation with this conditioning property can be modeled as a MAR, when it comes to inferring something about the underlying distribution. However, in compositional data, this becomes more complicated than the usual MAR concept in multivariate analysis. Assume a composition $(0.1, 0.4, 0.5)$ to be observed without the last value. We would get $(0.2, 0.8, \text{MAR})$, which looks quite different due to reclosing. On one hand, this is another hint that individual amounts cannot be analyzed without the rest of the composition. On the other hand, this implies that working with missing values in compositional data is much more complex than in classical data. While it is practically impossible to deal with these kind of missings in "`rcomp`" scale, for "`acomp`" scale, we have the projection approach briefly described in the preceding section.

The logistic regression approach of the preceding section could be used to distinguish MCAR from MAR. If we accept the Wald test there, we may say that the probability of missing is independent of the observed part, as an MCAR requires. Building on the example of the preceding section, these lines simulate a dataset with MCAR values on the fourth column:

```
> lostMCAR = runif(nsim)<0.3
> simMCAR = simDAT
> simMCAR[lostMCAR,4] = MARvalue
```

Now we can compare the logistic regression results in this case with those obtained with MAR:

```
> datset = data.frame(ilr(simMCAR[,-4]),
                        y = factor(is.MAR(simMCAR[,4])))
> lrmMCAR =  glm(y ~ V1 + V2 , family=binomial, data=datset)
> waldtest(lrmMCAR)

Wald test

Model 1: y ~ V1 + V2
Model 2: y ~ 1
  Res.Df Df    F Pr(>F)
1     97
2     99 -2 1.51   0.23
```

Thus, we can estimate the mean with the implemented projection approach:

```
> mean(simMCAR)

     A      B      C      D
0.1045 0.2099 0.3186 0.3670
attr(,"class")
[1] acomp
```

When imputing values, we could use the same approach as with MAR, using a regression, but no weighting is required.

7.2.5 Treating True and Structural Zeros

True zeroes in "ccomp" objects do not imply any special problem, as they have a fully meaningful quantitative value (0), resulting from a well-known stochastic model (the multinomial distribution, Sect. 3.1). On the other hand, the statistical model for a *structural zero* is much less clear. Strictly speaking, observations with and without the structural zero are qualitatively different; they belong to different subpopulations. Thus, it could be interesting to model the conditional distribution of the observed subcomposition depending on the presence/absence of structural zeros. However, if SZ appear in several variables or both the number of observations with and without SZ are not large enough, our estimations will not be reliable. It might be thus interesting to analyze the dataset in an integrated way, somehow filtering out the structural zeroes.

In general, imputation does not make much sense for structural zeroes, because a true value would be 0 or not existing, and thus a guessed positive value does not honor the truth. However, if we relax our aim and say that we would like to infer a value that would be observed if it could be defined, we implicitly assume the observations with SZ are similar to those fully observed, and thus the imputed value would follow the assumptions of MCAR.

For the "rcomp" scale, a structural zero should be understood as true zero and can be replaced by that value. For "acomp" and "ccomp" scales, the structural zero is rather an undefined value. However, in "ccomp" scale, one typically has true zeroes rather than structural ones. Graphical representation of zeroes does not entail any problem in ternary diagrams: they are points placed exactly on a border. In scatterplots of log ratios, however, zeroes fall at the infinity and cannot be represented as they are: one has to represent them like MAR values, as lines.

7.2.6 Treating Not Random Missings

Missing not at random (MNAR) does often not really mean that the missingness is not random. It rather means that the probability of being missed depends on the complete value, including the actual true value of the value reported as MNAR. It is also often suggested that MNAR components are not tractable at all. This is not strictly

true. The main difference to MAR values is that there is no possibility to infer from the data how the probability of missingness depends on the actual missing value.

A possible scenario for a missing not at random value is that a measurement procedure fails more likely for extremely high or low values. BDLs may be seen as special cases of MNAR values. That would lead to an underrepresentation of high or low value and thus a bias in the analysis. However, when we assume a (a priori stochastic or statistically inferred) law for the value to be missing, we could exploit that in Bayesian modeling.

Let $p_\theta(x)$ denote the probability of observing the critical component i in a composition and θ the unknown parameter with a prior P^θ. Further, let $f(x|\theta)$ denote the conditional distribution of the composition given the parameter and x_- denote a projection of the composition with a missing on the non-missing subspace. Then, the complete likelihood of the an observation without a missing component would be $L_x(\theta) = p_\theta(x) f(x|\theta)$ and the likelihood of an observation with a missing $L_x(x_-) = \int_{-\infty}^{\infty} p_\theta(x_- \oplus \alpha \mathbf{e}_i) f(x_- \oplus \alpha \mathbf{e}_i, \theta) \mathrm{d}\alpha$. This is computable if the conditional expectation of the missingness probability given all observed components is computable (analytically or numerically). Depending on the model, the missingness parameters may or may not be identifiable. Actually, p_θ as a function is identifiable if it is nonzero and it is allowed to do replicate measurements of the potentially missing component on the same sample.

If we assume θ is known (or previously estimated), an imputation could be done using the conditional density of the unobserved balances which is proportional to

$$f(\alpha|x_-, \theta) \propto (1 - p_\theta(p_\theta(x_- \oplus \alpha \mathbf{e}_i))) f(x_- \oplus \alpha \mathbf{e}_i, \theta)$$

resulting in a imputation $x_- \oplus \alpha \mathbf{e}_i$ for an α simulated according to that density. In an approximate frequentistic approach, we could, e.g., start from an external dataset observing the missingness probability as a function of the true value (with the variable settings of Sect. 7.2.3):

```
> simDAT2 = rnorm.acomp(nsim, mn, vr)
> obsProb <- function(X) exp(-0.8*X[,4]^2)
> Missing = obsProb(simDAT2) < runif(nrow(simDAT2))
> ExternalReferenceData <- data.frame(Missing=Missing,
                                      ilr(simDAT2))
> lrmMNAR<-glm(Missing~V1+V2+V3,
               family=binomial,data=ExternalReferenceData)
> waldtest(lrmMNAR)

Wald test

Model 1: Missing ~ V1 + V2 + V3
Model 2: Missing ~ 1
  Res.Df Df    F Pr(>F)
1     96
2     99 -3 1.76   0.16
```

We simulate a dataset with MNARs following the same probability law:

```
> simMNAR = simDAT
> simMNAR[obsProb(simMNAR)<runif(nrow(simMNAR)),4] = MNARvalue
```

We can now predict the missingness probability for each of the actually completely observed cases:

```
> tkObs <- is.NMV(simMNAR[,4])
> mp <- inv.logit(predict(lrmMNAR,
              newdata=data.frame(ilr(simMNAR[tkObs,]))))
```

Analogously to the MAR case, we can now obtain an estimate for the mean with nearly the same commands:

```
> w = 1/(1-mp)
> V = balanceBase(simMNAR, ~D/(A/(B/C)))
> ILR = ilr(simMNAR,V)
>  colnames(ILR)=c("V1","V2","V3")
> meanV1 = weighted.mean(ILR[tkObs,"V1"], w=w )
> bmean = mean(ILR)
> bmean[1] = meanV1
> ilrInv(bmean,V)

     A      B      C      D
0.1022 0.2053 0.3115 0.3810
attr(,"class")
[1] acomp
```

So the trick is just that we need to estimate probability of missing from a reference dataset, where the actual values of those parts that would be missing with our measurement procedure are not missing, e.g., because the corresponding samples are generated from standards or are remeasured with a better technology. The conditional distribution and conditional expectation could in principle be computed via Bayes' theorem from the probability of missingness and a known for the dataset. The analog for the imputation given for MAR values could be seen as a first approximation to that:

```
> w = mp/(1-mp)
> wrm = lm(V1~V2+V3, data.frame(ILR[tkObs,]), w=w )
> ILR[!tkObs,1] = predict
             (wrm, newdata = as.data.frame(ILR[!tkObs,]) )
> impMNAR = ilrInv(ILR, V=V)
> plot(ILR[,1], ilr(simDAT,V)[,1], col=is.MNAR(simMNAR[,4])+1)
```

which delivers a deterministic type 2, and

```
> ILR[!tkObs,1] = predict
        (wrm, newdata = as.data.frame(ILR[!tkObs,]) ) +
+   rnorm(sum(!tkObs), mean=0, sd = sqrt(var(wrm)) )
```

```
> impMNARrandom = ilrInv(ILR, V=V)
> plot(impMNARrandom,col=is.MNAR(simMNAR[,4])+1)
```

which delivers a random imputation which approximates type 3 imputation. Be, however, aware that all is based on the assumption that the probability of observing a datum is never 0.

A component is "missing not at random" in the best sense of the word, if the mechanism of missingness is not random, but something else like deterministic, or according to some plan. For instance, if the owner of a mine dropped all observation he did not want the buyer to see, because he wants to sell the mine. In this case, it is not possible to estimate the model or estimate an observation probability. To the authors' knowledge, there is no methodology to correctly regain the removed data.

7.2.7 Treating Values Below the Detection Limit

The most common version of missing values in compositional data is observations below the detection limit. This type of missing actually contains a quantitative information on the values: they are small and almost surely smaller than about 2 times the reported detection limit. The reader might still wonder: *was not the meaning of below the detection limit, that the value is below the detection limit?* Actually, it means that the obtained measure was not distinguishable from zero for the particular analytical technique used. If now the measured value is slightly below the detection limit, we may consider that the true value is certainly between zero and up to around 2 times the detection limit itself, as will be explained in Sect. 7.2.8.

However, all currently available approaches work with the hypothesis that "below detection limit" readings imply true values which are positive and below the known detection limit. Seen in this way, the actually observed vector has two parts: (1) a quantitative information on the observed subcomposition and (2) an upper bound on the ratios of the missing vs. the observed components.

Current approaches take an additive lognormal model. Based on such a model, it should be possible to compute a likelihood for either regular and "below detection limit" (BDL) observations. For a single BDL component, we can efficiently compute the probability of being BDL conditional to the observed subcomposition of the others Palarea-Albaladejo and Martìn-Fernández (2008). For multiple BDL components, a numerical computation is still possible. Having these probabilities available, a numerical maximum likelihood (ML) method or a MCMC Bayesian approach could be used to estimate the parameters of the model. Tjelmeland and Lund (2003) tried such an approach using a simple additive-logistic-normal model in a Bayesian framework including spatial dependence.

Expectation-maximization (EM) could be used here in the following way (Palarea-Albaladejo and Martìn-Fernández, 2008):

1. Start the algorithm by providing guesses for the BDL values, e.g., by some very simple deterministic imputation

2. Estimate the parameters of the additive-lognormal-based model (e.g., a compositional linear model) in the usual way by maximum likelihood
3. Impute the values as the conditional expectations of the values given being below the detection limit
4. Repeat from step 2 until convergence or iteration limit is reached

Since in every step there is a deterministic imputation, variances will be underestimated as discussed in Sect. 7.2.1.2. If the leverages of the imputed values in the linear model estimation step are all below 1, we would assume the algorithm to converge to unique optimum determined by the data, so that we can assume to have a stable model estimation, tending to overfit the data.

We again simulate a dataset with a detection limit of 0.05 in the first component. Recall that a BDL is encoded as the negative of its detection limit. The `simDAT` is defined above.

```
> simBDL = simDAT
> simBDL[simBDL[,1]<0.05,1] = -0.05
> simBDL = acomp(simBDL)
```

We now define a function which does a simple EM-algorithm iteration following the ideas of Palarea-Albaladejo and Martìn-Fernández (2008), but using an ilr- rather than an alr basis. The function is a substantial simplification because it does not treat missings in multiple variables. This function needs a balance base `V` having the first coordinate representing the balance of the missing component against all the others. `iter` gives the number of EM iterations.

```
> simpleEMBDLreplace <-function(data,V,iter=10) {
+   "Which rows have a BDL?"
+   tkBDL = is.BDL(data[,which(V[,1]>0)])
+   "Function to compute the conditional expectation in step 3"
+   CondExpectation = function(limit,cmean,sigma) {
+     x <- (limit-cmean)/sigma
+     cmean-sigma*dnorm(x)/pnorm(x)
+   }
+   "Initialising the iteration"
+   X <- ilr(zeroreplace(simBDL,a=1),V)
+   "Computing the detection limit in ilr"
+   dlVal <- X[tkBDL,1]
+   "EM-loop"
+   for(i in 1:iter) {
+     "Maximisation Step:"
+     Mean = mean(X)
+     Var  = var(X)
+     "Expectation step"
+     "1. Build conditional distribution"
+     beta = solve(Var[-1,-1],Var[-1,1])
+     cmean = c(beta) %*% t((X-Mean)[tkBDL,-1])+Mean[1]
```

```
+     SD  = c(sqrt(Var[1,1]-c(Var[1,-1])%*%c(beta)))
+     "2. Compute conditional expectation"
+     CE  = CondExpectation(dlVal,cmean,SD)
+     "3. Show covergence"
+     cat("Iteration ",i," step=",norm(X[tkBDL,1]-CE),
                              " mean=",ilrInv(Mean,V),"\n")
+     "4. Do the actual imputation"
+     X[tkBDL,1]=CE
+     X <- rmult(X)
+     impBDL = ilrInv(X,V,orig=data)
+   }
+   "Store estimated Mean and Variance"
+   attr(impBDL,"mean")<-ilrInv(Mean,V,orig=data)
+   attr(impBDL,"var")<-ilrvar2clr(Var,V,data)
+   "Return results"
+   return(impBDL)
+ }
```

The reader can omit the lines with only strings, which are for documentation. Finally, we compute the basis isolating the first component and call this ad hoc function:

```
> V <- balanceBase(simBDL,~A/(B/(C/D)))
> erg <-simpleEMBDLreplace(simBDL,V)

Iteration  1  step= 2.11  mean= 0.1196 0.1931 0.2931 0.3942
Iteration  2  step= 0.6658  mean= 0.1061 0.1961 0.2975 0.4003
(...)
Iteration  9  step= 0.0006238  mean= 0.1 0.1974 0.2996 0.403
Iteration  10  step= 0.0002328  mean= 0.1 0.1974 0.2996 0.403
```

The iteration fast approaches a stable estimate.

A deterministic imputation without previously estimated parameters, e.g., by replacing each BDL value by $2/3L$ can produce outliers (see Sect. 7.3) in different directions. The actual ratios can be underestimated or overestimated. It might thus seem a good idea to use robust estimation in the subsequent analysis. However, robust methods do not present a consistently better performance. Figure 7.2 shows a simulated example situation. We have three components $\{A, B, C\}$ where the A/B ratio is practically fixed to 99:1, A and C are always observed, while B is often below the detection limit of $L = 0.005$. The generating code follows:

```
> set.seed(27)
> logit <- function(x) log(x/(1-x))
> logitI<- function(x) exp(x)/(1+exp(x))
> observe <- function(x,L=0.005,I=L*2/3) {
+   x<-x+rnorm(length(x),0,L/3);
+   ifelse(x>L,x,I)
```

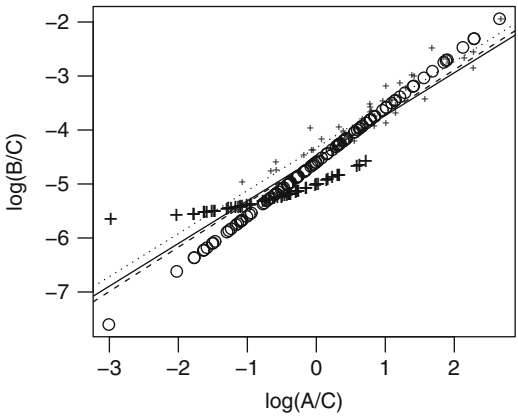


Fig. 7.2 Balance scatterplot illustrating bad behavior of deterministic imputation, even with robust methods. *Small crosses* are observed values, *large crosses* involve an imputation, *circles* are the true values, and *lines* show inferred regression based on imputation (*solid* = least squares; *dashed* = robust; *dotted* = without imputed values)

```
+ }
> x <- logitI(seq(logit(0.1),logit(0.9),by=0.1))
> x <- logitI(rnorm(100,0,1))
> A <- 0.99*x
> Aobs <- observe(A)
> B <- 0.01*x
> Bobs <- observe(B)
> C <- 1-x
> Cobs <- observe(C)
> Total <- Aobs+Bobs+Cobs
> Aobs = Aobs/Total
> Bobs = Bobs/Total
> Cobs = Cobs/Total
> plot(log(A/C),log(B/C))
> points(log(Aobs/Cobs),log(Bobs/Cobs),
                  pch="+", cex=ifelse(Bobs<0.005,1,0.5))
> abline(lm(log(Bobs/Cobs)~log(Aobs/Cobs)))
> require(robustbase)
> abline(lmrob(log(Bobs/Cobs)~log(Aobs/Cobs)),lty=2)
> abline(lm(log(Bobs/Cobs)[Bobs>0.005]~log(Aobs/Cobs)
                                    [Bobs>0.005]),lty=3)
```

Figure 7.2 shows a deterministic imputation on a simulated data set, where a clear regression $A = 0.99 * (1 - C)$, $B = 0.01 * (1 - C)$, $\text{logit}(C) \sim N(0, 1)$ is found, and all components are observed with a detection limit of $L = 0.005$ (i.e., with an additive standard error of 0.005/3) and are imputed at (2/3)L. Circles show the balances computed from the true values. Crosses show the balances computed from the observed and imputed values. Small crosses are completely observed, while large crosses are those data points involving imputations. The true regression would correspond to a straight line following the circles.

As a rule of thumb, one might say: if the number of BDLs is low, any of the methods will work considerably well. If the number of BDL is substantial, nothing is safe.

7.2.8 The Physical Meaning of a Detection Limit

Geochemical analysis and analytical chemistry traditionally work with an additive error assumption, rather incompatible with the Aitchison geometry. It is important to understand the reasons of this choice and the actual meaning of a detection limit to adequately handle it in a compositional analysis. This section is a brief introduction to these issues.

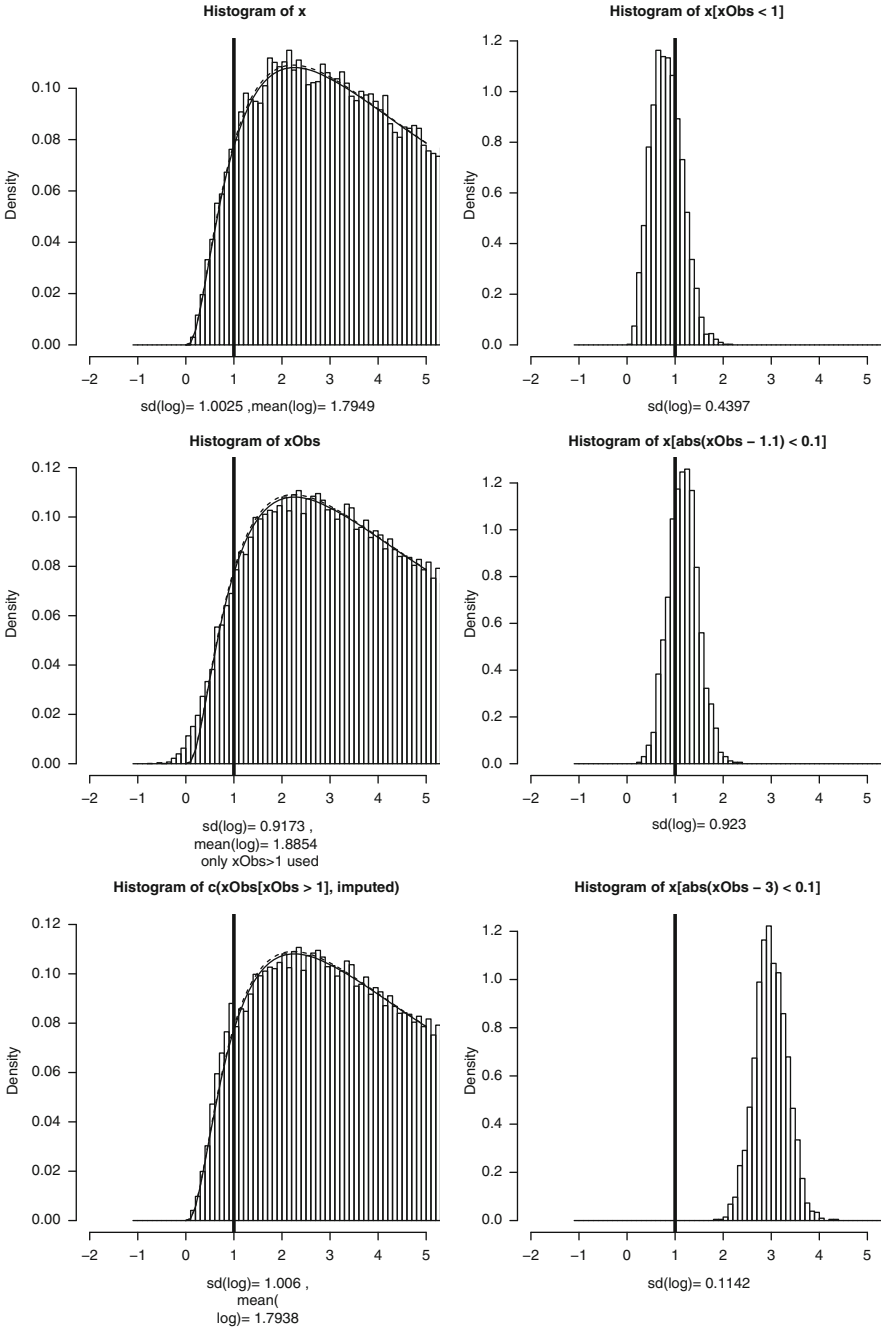
In a geochemical analysis, a sample of more or less fixed size is taken, crunched, and mixed. Afterwards, smaller subsamples of fixed, but not exactly controlled size are taken, further processed, and finally subject to some procedure. Each subsample provides a reading which is proportional to the actual content of one or the other component. Typical readings might be ion or isotope counts hitting a sensor.

The errors in the readings come from various sources:

- A multiplicative error from selective loss of mass during crunching and mixing
- A multiplicative error from the size of the subsample taken
- A multiplicative error from loss of mass or solution during processing the subsample
- An error nonlinearly depending on some other components, e.g., interactions in the effect of the laser ablation in chromatography
- An additive error in the reading, e.g., due to random background noise and Poisson counting errors. Near zero, the additive error dominates the other error sources

While the multiplicative errors nicely fit into the "`acomp`" and "`aplus`" approaches of compositional data analysis, the additive error has always caught the main attention of the analysts: e.g., measuring an amount in a machine like gas chromatograph has a characterized measurement error induced by additive noise added to the readings. This error has not a zero mean, which would thus always lead to an overestimation of the actual amount. Thus, its expectation, typically estimated from the background readings before and after the sample is processed, is subtracted from the actual intensity reading values. The difference is afterwards multiplied with a proportionality constant, which rescales intensities or counts to desired amounts or portions.

If the error is small with respect to the actual value, this results in a small relative error. However, when we measure a small amount, possibly zero or near to zero, the variability of the additive noise dominates the measurement. It is quite likely (about half of the cases) that the subtraction leads to a negative value, and obviously, we cannot provide a numerical value for those amounts. Also if the true amount is exactly zero, then with probability $1/2$, the difference is a positive value somewhere between zero and two standard deviations of the additive noise. This is conveyed by Figs. 7.3 and 7.4.


Histogram of x
Density
sd(log)= 1.0025 ,mean(log)= 1.7949
Histogram of x[xObs < 1]
Density
sd(log)= 0.4397
Histogram of xObs
Density
sd(log)= 0.9173 ,
mean(log)= 1.8854
only xObs>1 used
Histogram of x[abs(xObs – 1.1) < 0.1]
Density
sd(log)= 0.923
Histogram of c(xObs[xObs > 1], imputed)
Density
sd(log)= 1.006 ,
mean(
log)= 1.7938
Histogram of x[abs(xObs – 3) < 0.1]
Density
sd(log)= 0.1142

In geochemical analytical techniques, one traditionally does not think in terms of compositional data, but rather in terms of proving the presence of a given component. Then, it would not make sense to give any value if the measurement is so low that it could well be compatible with the background. Thus, we can assume that nothing of this component is actually there. It will thus be reported as *below detection limit* and meant to be (almost) indistinguishable from zero.

On the other hand, from the perspective of a compositional geometry things look a little worse. For a reading below the detection limit, an actual concentration of zero is an explanation as reasonable as something two standard deviations above the detection limit. This uncertainty corresponds to an infinite relative error. But even worse, the same conclusion is valid for observations slightly above the detection limit. Let L denote the detection limit. Then, by a reading of $X = L + \varepsilon$, $\varepsilon > 0$, we only show that the value is with high probability in the interval

$$[X - L, X + L] = [\varepsilon, 2L + \varepsilon],$$

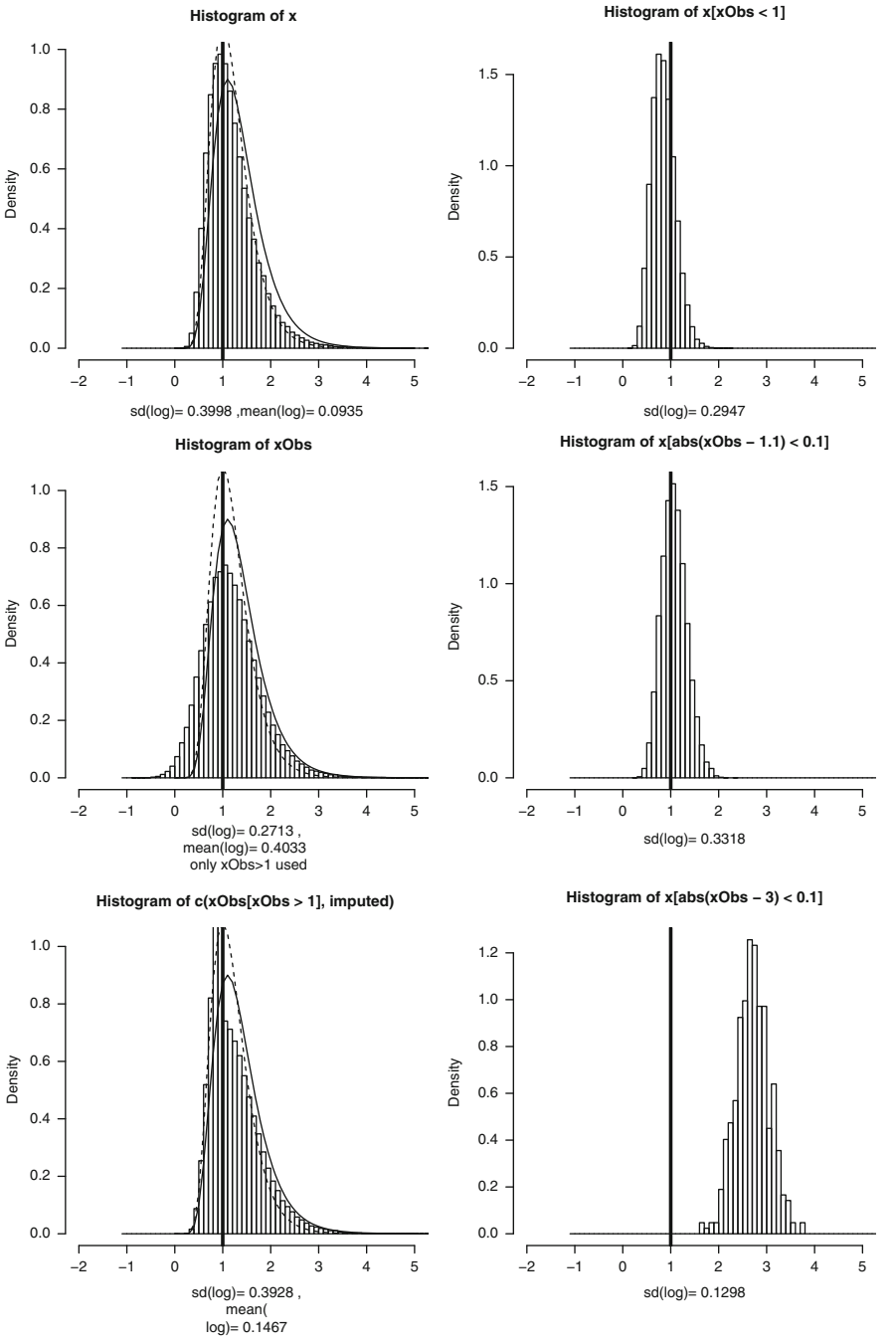
which corresponds to a relative error up to

$$\frac{X}{X - L} = \frac{L + \varepsilon}{\varepsilon} = \frac{L}{\varepsilon} + 1.$$

This can take any value between 1 and ∞. The relative error is thus uncontrolled for values slightly above the detection limit. Furthermore, if the true concentration is actually between L and $2L$, there is a fair chance that the value is actually reported as below detection limit.

Looking more closely from the point of view of a Poisson counting process, counting errors increase with increasing concentrations. Using the notation of x for the true value and assuming that the background variability is also a Poisson counting error with standard error $L/3$, then $x = L$ would have the same standard deviation from the counting error and thus

Fig. 7.3 Interplay of additive error, detection limit, conditional distribution, and lognormal imputation in a well-behaved case. The variable x denotes the true amount value distributed according to a lognormal $LN(\ln(6), 1)$ distribution (i.e., corresponding to an "`aplus`" object, while *xObs* is a measured reading scaled to the corresponding x values (i.e., $X + \varepsilon$, $\varepsilon \sim N(0, 0.5)$. The figures on the *left* show the histograms of x, *xObs* and a dataset of reported and imputed values. The *vertical thick line* shows the detection limit. The *lines* show a lognormal distribution fitted to the *xObs* values above detection limit by maximum likelihood (ML, *solid line*) and by the EM-algorithm (*dashed line*; see Sect. 7.2.7 for details). Surprisingly, both fit well to the x values and not to the *xObs* values below the detection limit. However, a random imputation using this model creates a strange misfit in the produced densities. This happens because the portion of values actually below the detection limit is higher than the lognormal model predicts. The figures on the *right* show conditional distributions of (a) x given observations of values below the detection limit, (b) slightly above detection limit, and (c) above 3 times the detection limit. The distributions show clearly different variabilities on "`aplus`" scale, demonstrating the high relative error of values observed near the detection limit


Histogram of x
Density
sd(log)= 0.3998 ,mean(log)= 0.0935
Histogram of x[xObs < 1]
Density
sd(log)= 0.2947
Histogram of xObs
Density
sd(log)= 0.2713 ,
mean(log)= 0.4033
only xObs>1 used
Histogram of x[abs(xObs – 1.1) < 0.1]
Density
sd(log)= 0.3318
Histogram of c(xObs[xObs > 1], imputed)
Density
sd(log)= 0.3928 ,
mean(
log)= 0.1467
Histogram of x[abs(xObs – 3) < 0.1]
Density
sd(log)= 0.1298

$$\text{var}(X) = \frac{L^2}{9}\frac{x}{L} + \frac{L^2}{9} = \frac{1}{9}(Lx + L^2),$$

or also

$$3\text{sd}(X) = \sqrt{Lx + L^2}.$$

This corresponds to a relative error of

$$r = \frac{3\text{sd}(X)}{x} = \sqrt{\frac{L}{x} + \frac{L^2}{x^2}},$$

which is a decreasing function of x, infinite for x tending to 0, and still $\sqrt{2} \approx 1.41$ for $x = L$ i.e., 140 % of error on top of the multiplicative error, decreasing to, e.g., $\sqrt{1/10 + 1/100} \approx 0.33$ for $x = 10L$. Clearly if the background has more variability than the mere counting error, we would get smaller errors. If no additional counting error is generated, the relative error function would be

$$r = \frac{3\text{sd}(X)}{x} = \frac{L}{x}.$$

In all cases, the relative measurement error is due to not constant additive error component.

In a Bayesian approach, a BDL with additive error needs a two-step modeling approach, where each observation is a hidden true value according to the modeled distribution plus an additive normal error with mean zero and standard deviation according to the precision of the measurement procedure. The likelihoods of both observed data and BDL values according to this mixture model has no closed analytical form and must be approximated with MCMC methods.

Fig. 7.4 Interplay of additive error, detection limit, conditional distribution, and lognormal imputation in an example where imputation based on a lognormal model is very problematic. The variable x denotes the true concentration value distributed according to a lognormal $LN(\ln(1.1), 0.4)$ distribution, e.g., in an "`aplus`" geometry. In contrast with Fig. 7.3, here the behavior of the observed values $xObs = X + \varepsilon$, $\varepsilon \sim N(0, 0.5)$ above the detection limit is qualitatively different from that of the true values, and the portion of values below detection limit is very poorly explained by the lognormal distribution model. The *lines* show a lognormal distribution fitted to the *xObs* values above detection limit by ML (*solid line*) and by the EM-algorithm (*dashed line*). Consequently, the values imputed following the conditional distribution in the fitted model will exhibit a strange behavior. The reason for such bad performance of the imputation can be seen on the right-hand side, showing conditional distributions of x values in three cases: (a) observations below detection limit, (b) slightly above the detection limit, and (c) above 3 times the detection limit

7.3 Outliers

7.3.1 Background on Outliers

There is no absolute mathematical definition of an outlier. Loosely speaking, an outlier is an atypical observation outside the main mass of the dataset. Most compositional models are formulated in terms of additive lognormal models or their multivariate normal counterparts in coordinates. Outliers can thus be detected with the classical affine equivariant methods for multivariate outlier detection, applied to the alr or ilr transformation of the dataset (Filzmoser and Hron, 2008).

It is well known (Rousseeuw and Leroy, 2003, e.g.) that outliers do not easily show up in the direct residuals of non-robust linear models. It is thus of the utmost importance to use robust estimation techniques to detect outliers.

It might be difficult to decide which is the most adequate estimator when outliers are present. A robust estimator would fit the model to the "non-outlying" part of the data. This may be what we want, when the outliers are deemed as observational errors or otherwise unrelated to the process of interest. For instance, one may be interested in a natural process of enrichment, while the outlying compositions have been generated by sampling or lab treatment. In this case, our main aim is to make the analysis independent of these "wrong" observations. However, this is not always the case. Extremal observations might be important parts of the natural process. The most typical example of this second situation is in mining, where areas of extremal enrichment are in fact those of interest, maybe even representing the difference between having just an anomaly or a deposit worth being mined. An illustration of this problem can be found in Fig. 7.5. It shows a comparison of classical and robust estimation methods of the mean. Both histograms show of 10,000 means of 30 standard normal variables mixed with a 1/15 probability with a normal variable with mean and standard deviation of 20. The first histogram uses classical means and the second trimmed means for a robust estimate. Neither the robust nor the classical gives a useful estimate of the overall expectation, marked by the vertical line. In such a case, leaving out the "outliers" would lead us to a flawed conclusion. Obviously a simple model, e.g., an additive logistic normal, would not model these outliers, and the meaning and sense of the associated estimators would thus be questionable. A parameter is defined with respect to a model. Thus, if the model is known to be wrong, how can we meaningfully estimate the parameter?

One way to look at the problem would be to consider parameters valid for large classes of models, e.g., moments (expectation and variances). But in the mining example above, the Aitchison expectation of a composition is not the essential parameter. The economically essential parameter is the mean grade of the deposit. Likewise for studies of pollutants, it is often not important to model the distribution of the compositions of the main part of the data. Much more relevant is the distribution and portion of compositions with pollutants over legal

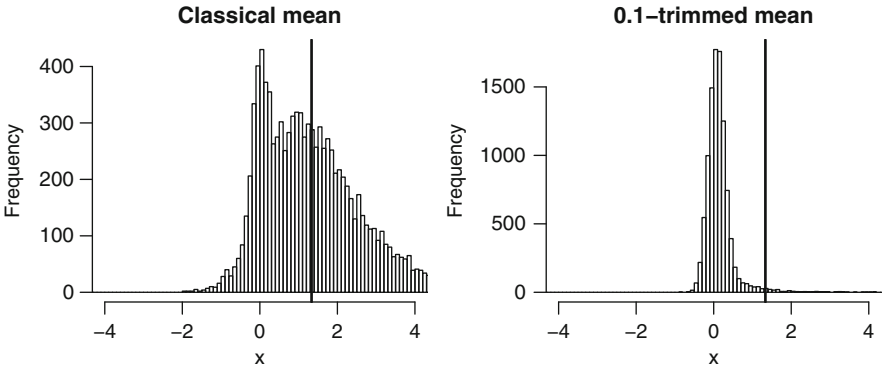


Fig. 7.5 Comparison of classical and robust estimation methods of the mean in the case of a mixed population. The *vertical line* represents the true mean

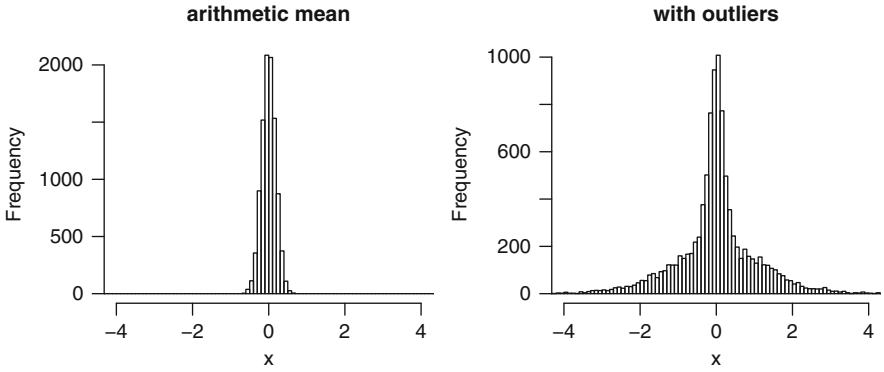


Fig. 7.6 Effect of outliers on dispersion of arithmetic average estimates of a normal population. The first histogram shows 10,000 means of 30 standard normal variables each. The second histogram shows the 10,000 means of the same situation where each observation is a wrong measurement with a probability of 1/30. These wrong measurements are again modeled by a zero-mean normal random variable, but now with a standard deviation of 30

limits. This portion might be well described by quantiles if the compositions have a lognormal distribution, but this will not be the case if a relevant portion of outliers with respect to that model is present.

Even if Aitchison expectation and variance are the variables of interest, outliers make the situation problematic. If we use classical unbiased estimators (i.e., log-ratio-based mean and variance), these estimators are strongly influenced by the extreme observations. That is not unexpected in the first place, because a heavy tail in the theoretical distribution also has a strong influence on the theoretical expectation. However, precision and reliability of these estimates decrease under the presence of such extreme observations, even when showing up with low probability. Figures 7.6 and 7.7 show such an effect on raw and log scale. Note that estimates

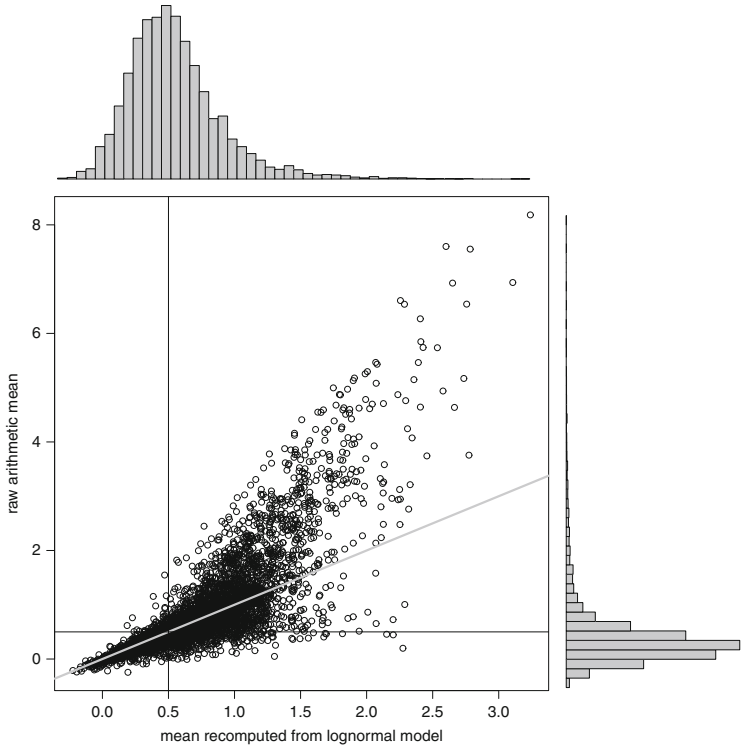


Fig. 7.7 Effect of outliers on dispersion of arithmetic average estimates (in log scale) of a lognormal population. The diagram shows 10,000 arithmetic means of 30 standard lognormal variables each, where each observation is a wrong measurement with a probability of 1/30. These wrong measurements are again modeled by the same lognormal random variable, but now with a standard deviation of 3. *Upper histogram* and *horizontal axis* show the estimate obtained using the lognormal expectation formula $\exp(\hat{\mu} + (1/2)\hat{\sigma}^2)$. *Right histogram* and *vertical axis* show the raw arithmetic estimate. Note the heavy tail towards the right produced by the presence of large outliers, especially for the classical arithmetic mean

computed with lognormal theory appear slightly more robust than classical raw arithmetic means, though this comes with the price of having to estimate a variance and may utterly fail if the assumption of lognormality does not hold. With an additive-logistic-normal model, the effect would be comparable to this lognormal case. Note that geometric mean and compositional center would actually behave rather like the arithmetic mean under normality assumption.

Thus, there is no direct, easy, and general way to work with outliers. Outliers are defined with respect to models. Outliers have different interpretations and implications depending on their origin and the question. Outliers suggest errors in the measurement process, but do not offer conclusive evidence on that. They may also suggest that our model assumptions are utterly wrong: we may assume a homogeneous sample and find many clustering outliers, thus hinting to the presence

of special subgroups; or we may assume additive logistic normality when our tails are heavier than that, just producing huge amounts of nominal outliers on these tails.

7.3.2 Types of Outliers

From a phenomenological point of view we can distinguish the following types of outliers:

- *Extreme observations*
  Extreme observations might happen just by chance in random observations. As an intuitive illustration, recall that we do not expect a value of -7 as outcome of a univariate, standard normal distribution: however, this occurs—seldom indeed, with a very low probability, but it is not impossible. Thus, extreme observations are characterized by being scarce and become even rarer the farther we depart from the main part of the dataset.
- *Plain errors*
  Errors may occur for a variety of reasons, most typically in the measurement process or in the transmission/copy. For instance, a change of units may be forgotten or the decimal point misplaced. The characteristic of this type of errors is that they typically affect one single variable, the other variables should be "correct," falling within the main body of the data set in the subcomposition without the error-affected variable.
- *Individual atypical samples*
  Some individuals, though being "correct" (without errors) might be too atypical with respect to the main body of the dataset, e.g., coming from very rare subpopulations with contrasted properties. The presence of many atypical outliers is a sign that the sampling did not cover just one homogeneous population.
- *Polluted samples*
  Compositional data, in particular geochemical data, may suffer pollution, mostly due to an inadequate treatment during sampling or in the lab. These kind of irregularities may affect a significant portion of the dataset. Characteristically, the polluted samples should roughly follow a segment from the typical portion of the data to the pollutant composition and be bounded between these two end-members.
- *Heavy-tailed distributions*
  Actual data rarely exhibit a nice, normal distribution. In practice, one often encounters heavy-tailed distributions, even in log scale. In these cases, when assessing atypical values with a normal model in mind, we will necessarily detect several atypical observations with no clear structure.
- *Multimodality*
  Finally, a different form of non-normality is detected by the presence of several modes, representing several subpopulations. In this case also fall some special artifacts typical of compositions, like apparent subgroups produced by replacing many BDL values by a fixed value.

This is not a complete list of possible causes for atypicality. Nevertheless, since some visualization tools of this classification are provided (see Sect. 7.4.4), the analyst can see if a given sample or group of samples does not satisfy any of these patterns and look then for possible alternative causes. Most of these types of outliers can also be found in conventional multivariate datasets.

7.3.3 Robust Estimation

Under the presence of outliers, one has to use robust estimation procedures. An estimator is considered non-robust if its value can arbitrarily change by simply perturbing one single datum (i.e., an outlier). On the contrary, an estimator is called *robust* with *breakpoint* $b \in [0, 1/2]$ if the estimated value does not arbitrarily change if the portion of outliers in the dataset is below b. The "`compositions`" package allows to use different robust estimators to estimate mean and variance. They can be selected by the *robust option* or the `robust=` parameter of the `var` command. Computation in the preceding examples of the book are done with

```
> options(robust=FALSE)
```

using the usual non-robust estimators for mean and variance. A simple robust estimator is the *minimum covariance determinant* estimator, or *MCD* (Rousseeuw and Leroy, 2003), typically used in multivariate data analysis. This algorithm finds the mean vector and variance matrix of the typical subgroup of the data by finding an ellipsoid with minimum volume covering a $1 - b$ portion of the dataset. The center of the ellipsoid is used as an estimate for the mean vector. The variance matrix is estimated as the symmetric positive definite matrix describing the algebraic equation of this ellipse, scaled in such a way that the ellipse would contain a probability of $1 - b$ of a multivariate normal distribution. The fast minimum covariance determinant estimator for real multivariate data is available in **R** in the `robustbase` package by the `covMcd` command.

7.3.4 Detection and Classification of Outliers

Before outliers can be treated in any way, they need to be identified and classified. Different types of outliers have different implications for the further statistical analysis. Wrong data values can be corrected or removed from the analysis. For individual atypical values and polluted samples, we might want to understand their cause. In case of outliers originating from heavy-tailed distributions, multiple modes, or other kinds of multiple subpopulations, we need to perform any subsequent analysis aware of that more complicated structure: for instance, we might use a non-normal distribution to describe the dataset. The key problem of outliers is thus identification and classification.

Identification of outliers is not an easy task. Outliers do not come self-identified like missing values, and we have to find a criterion to highlight their presence. A typical one is to compare the spread of the data with the one expected for a normal distribution. The usual way to detect outliers is to compute the Mahalanobis distance of each observation, better if it is based on robust estimates of center and spread. Rousseeuw and Leroy (2003, pp. 266–270) propose to compare this Mahalanobis distance to the $1-\alpha$ quantile of the appropriate χ^2-distribution and take as outliers all samples above the corresponding quantile level. In the simplex, a $D-1$ dimensional space, the appropriate degrees of freedom are $D-1$. Alternatively, we later propose to use as testing distribution the maximum of the Mahalanobis distance, which can be obtained via Monte Carlo simulations. This option is available in the package and will be discussed in Sect. 7.4.3.

Classification of outliers is a vital point to decide what to do with them, because removing an error outlier or an atypical single observation may be sensible, but directly removing data affected by any of the other four cases of atypicality (extreme observation, contaminated samples, heavy tails, and subpopulations) is an oversimplification. Extreme observations should be kept, as there is "nothing wrong" with them. Subpopulations and contaminated samples should be studied separately from the main population, but not discarded as errors: they might be as interesting as the main part, if not even more. Finally, outliers generated by heavy tails rather point out to the fact that the underlying distribution is not a normal one, also for the main part of the data. Although we may not remove these "not wrong" outlier, such deviations from normality will increase estimation errors and challenge optimality and correctness of all normal-based methods.

Some of these atypicality types can be detected with the help of a series of criteria based on the geometric structure of the simplex. For instance, an error outlier should look completely normal in the subcomposition without the erroneous part. Subpopulations should be identified with a cluster analysis based on the Aitchison distance (2.5). Finally, if we express each datum as its direction from the mean (as a unitary vector in the Aitchison geometry of the simplex or in the classical real geometry, compatible with an "`rcomp`" scale), then polluted samples should show up as distinct angular clusters.

7.4 Descriptive Analysis of Outliers

7.4.1 Robust Estimation of Mean and Variance

Mean and variance of the typical group are estimated using a robust estimator. With `robust=TRUE`, the "`compositions`" package uses the `covMcd` estimator on the `cdt`-transformed dataset to estimate the variance. The corresponding estimated center is stored as attribute `center` of the resulting variance estimate. There are other robust estimators available. They can be selected by the *robust option* or the `robust=` parameter of the `mean` and `var` commands.

```
> GeoChemSed=read.csv("geochemsed.csv",header=TRUE,skip=1)
> x = acomp(GeoChemSed,c("Cr","Zn","Pb"))
> mean(x)

 Cr     Zn     Pb
0.1508 0.5773 0.2719
attr(,"class")
[1] acomp

> mean(x,robust=TRUE)

    Cr     Zn     Pb
0.1647 0.5243 0.3110
attr(,"class")
[1] acomp

> var(x)

          Cr        Zn        Pb
Cr  0.09954 -0.03896 -0.06058
Zn -0.03896  0.08360 -0.04464
Pb -0.06058 -0.04464  0.10522

> var(x,robust=TRUE)

          Cr        Zn        Pb
Cr  0.09733 -0.03884 -0.05849
Zn -0.03884  0.06427 -0.02543
Pb -0.05849 -0.02543  0.08392

> variation(x)

       Cr     Zn     Pb
Cr 0.0000 0.2611 0.3259
Zn 0.2611 0.0000 0.2781
Pb 0.3259 0.2781 0.0000

> variation(x,robust=TRUE)

       Cr     Zn     Pb
Cr 0.0000 0.2393 0.2982
Zn 0.2393 0.0000 0.1991
Pb 0.2982 0.1991 0.0000

> options(robust=TRUE)
> var(x)

          Cr        Zn        Pb
Cr  0.09733 -0.03884 -0.05849
Zn -0.03884  0.06427 -0.02543
Pb -0.05849 -0.02543  0.08392

> options(robust=FALSE)
```

7.4.2 Detecting Outliers

Being an outlier is a soft concept. To get the concept operative, we need to give a strict mathematical definition. Unfortunately, whatever definition we choose, it will not be the same as the intuitive concept of an outlier as an observation somehow out of the main part of the dataset. The "`compositions`" package considers two of these definitions. Both definitions are based on the robust Mahalanobis distance of the composition to the robust center of the dataset. Mean and variance of the ilr-transformed dataset are robustly estimated with fast implementation of the minimum covariance determinant estimator in the `robustbase` package (Rousseeuw et al., 2007). Call these estimates $\hat{m}_r$ for the robust mean estimate and $\hat{\Sigma}_R$ for the variance estimate. Then, the robust Mahalanobis distance $M_r(x_i)$ of a composition x_i from the robust mean is given by

$$M_r(x_i) := (\text{ilr}(x_i) - m_r)^t \hat{\Sigma}_R^{-1} (\text{ilr}(x_i) - m_r)$$

The two definitions are:

1. A value x_i is classified as a suspected outlier at significance level α if $M_r(x_i) \geq \chi^2_{D-1,1-\alpha}$, where $\chi^2_{D-1,1-\alpha}$ denotes the $1-\alpha$ quantile of the χ^2-distribution with $D-1$ degrees of freedom. This distribution is the distribution of the true (i.e., using true mean and true variance) of an additive-logistic-normally distributed data. This approach is the direct generalization of the outlier detection procedure proposed by Rousseeuw and Leroy (2003) to compositional data done by Filzmoser and Hron (2008).
2. A value x_i is classified as proven outlier at significance level α if we reject the hypothesis that x_i belongs to a normal sample. That is, it classifies as outlier if $M_r(x_i) \geq k_{D,n}$, where $k_{D,n}$ is the critical value chosen in a way that the probability of classification of any of the observations as an outlier in a additive-logistic-normal random sample is α. The critical value $k_{D,n}$ is computed by a simulation algorithm.

Both approaches try to classify a composition as an outlier if it is too far from the center to be one of a sample of additive-logistic-normally distributed observations. However, the first approach does neither consider that the Mahalanobis distance is estimated nor the fact that each observation is tested, and thus, multiple (stochastically dependent) tests are performed.

Figure 7.8 shows examples of these classifications for six example datasets generated with the commands:

```
> data(SimulatedAmounts)
> par(mfrow=c(2,3),mar=c(1,1,1,1))
> outlierplot(acomp(sa.outliers1),type="scatter",
                                    class.type="outlier")

⋮

> outlierplot(acomp(sa.outliers6),type="scatter",
                                    class.type="outlier")
```

The types of outliers exemplified are:

1. `sa.outliers1`: No outliers. Data generated according to a normal distribution in the simplex.
2. `sa.outliers2`: About 10 % of the data has large errors in the first component.
3. `sa.outliers3`: One single outlier.
4. `sa.outliers4`: Data is generated with a heavy-tailed Cauchy-type distribution.
5. `sa.outliers5`: About 10 % of the data is polluted with a uniform random portion of material with a different composition.
6. `sa.outliers6`: 20 points of a second subpopulation with a different mean are added.

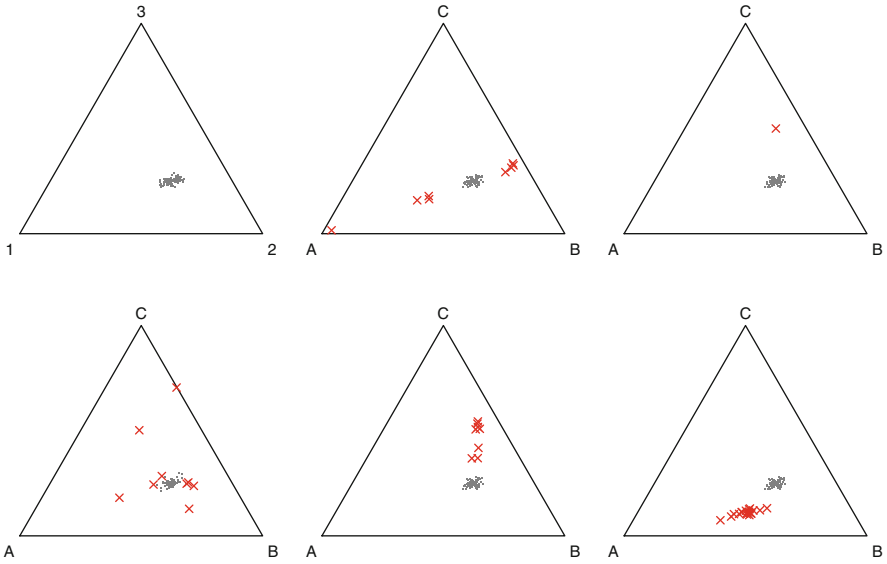


Fig. 7.8 Ternary diagrams of proven outliers according to a second outlier criterion. *Points* corresponds to observation not classified as outliers, *crosses* to proven outliers

The observations are classified with a command like

```
> OutlierClassifier1(x,type="outlier")

 [1] ok      ok      ok      ok      ok      ok      ok
(...)
[43] ok      ok      ok      ok      ok      ok      ok
[50] ok      outlier ok      ok      ok      ok      ok
[57] (...)
[85] ok      ok      ok
Levels: ok outlier
```

The command returns a factor showing the classification of each composition. If the type of the classification is set to `"outlier"` it only reports proven outliers. If the command is set to `"grade"` for grade of "outlyingness," the factors gets the additional level "extreme" for extreme observations or suspect outliers according to the first criterion:

```
> OutlierClassifier1(x,type="grade")

 [1] ok      ok      extreme ok      ok      ok      extreme
 [8] ok      ok      extreme ok      ok      ok      ok
(...)
[78] ok      extreme extreme extreme ok      ok      extreme
[85] ok      extreme ok
Levels: ok extreme outlier
```

Here "extreme" describes compositions outlying according to criterion 1 only, while "outlier" describes composition which are outliers with respect to criterion 2. "extreme" observations are only reported with the "grade" type of outlier classification.

7.4.3 Counting Outliers

It is difficult to prove that an individual observation is an outlier. Only observations quite far from the center are classified as proven outliers. On the other hand, in all large datasets, we will find extreme observations classified as suspected outliers according to the first criterion. However, for the sake of reporting outlier problems or exploring the reasons for outliers, it might often be an important step to find out how many outliers are in the dataset. For this, we can compare the empirical distribution function of the Mahalanobis distances with its distribution under the hypothesis that the dataset additive logistic normally distributed. If the empirical distribution function is below a lower predictive limit of that distribution, we are sure that there are too many values above that limit. And knowing how much below that limit the data fall, we can also find a lower bound for the number of data above that limit, i.e., a minimum number of expected outliers. This result is plotted in a specific type of outlier plot (Fig. 7.9). This figure is produced by the commands:

```
> opar<-par(mfrow=c(2,3), mar=c(3,3,1,1), oma=c(3,3,0,0))
> data(SimulatedAmounts)
> outlierplot(acomp(sa.outliers1),type="nout")
> outlierplot(acomp(sa.outliers2),type="nout")
> outlierplot(acomp(sa.outliers3),type="nout")
> outlierplot(acomp(sa.outliers4),type="nout")
> outlierplot(acomp(sa.outliers5),type="nout")
> outlierplot(acomp(sa.outliers6),type="nout")
> mtext(side=1, outer=TRUE, text="Mahalanobis distance")
> mtext(side=2, outer=TRUE, text="number of outliers")
```

For each plot, we can take the maximum of the curve, which provides us with a minimum for the number of outliers in the dataset. In these illustration datasets, we have therefore at least 0, 8, 1, 9, 17, and 63 outliers, showing that the method should not be trusted for substantial portions of outliers, while it works fine for datasets 1–4.

We apply now this methodology to a real dataset (see Fig. 7.10):

```
> par(mfrow=c(1,2), mar=c(4,4,1,1))
> x = acomp(GeoChemSed,c("Cr","Zn","Pb"))
> outlierplot(x,type="scatter",class.type="outlier")
> outlierplot(x,type="nout")
```

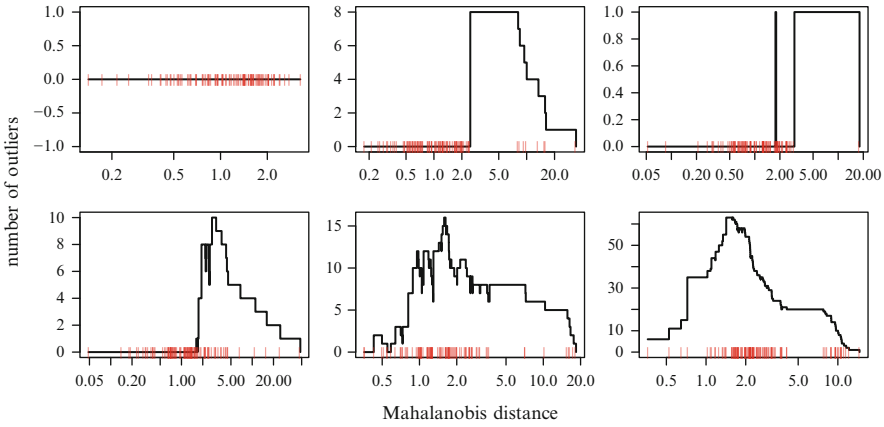


Fig. 7.9 Histograms of expected outliers above a given Mahalanobis distance threshold. A rug (red) represent the robust Mahalanobis distances of the data points. The *black line* shows how many outliers should be present (i.e., significantly proven) above that Mahalanobis distance

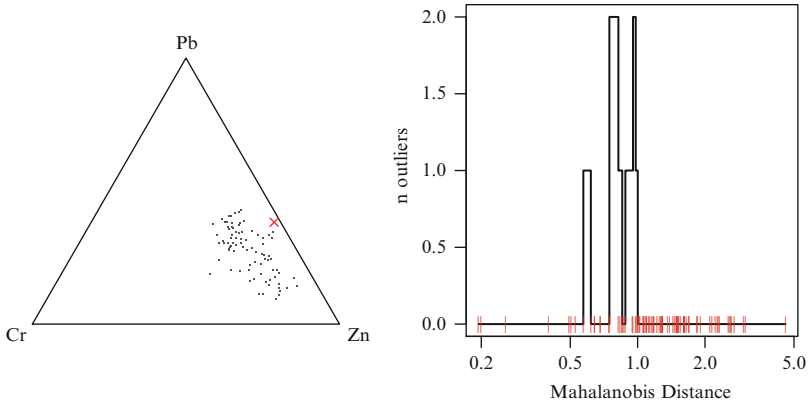


Fig. 7.10 Outlier detection in the geochemical dataset. Plots show type and number of outliers

These figures show that we can prove the presence of two outliers, but only one of them can be identified as a proven outlier. Thus, even if we remove that outlier, we still need to work with robust methods. On the other hand, there is no hint for multiple wrong values.

7.4.4 Classifying Outliers

If outliers are detected, we might want to investigate their genesis. There is no statistical procedure that positively identifies the reason of an outlier. However, we can exclude or guess reasons by identifying or disproving possible explanations

for an outlier. An important mechanism of creating outliers is a measurement error in a single component. Thus, an important tool is to check whether or not an outlier can be explained by a single component error. To this aim, we check whether the observation is still extreme, if that single component is removed from the composition. The following command does that for every subcomposition and returns a factor classifying the observations:

```
> OutlierClassifier1(x,type="type")

 [1] ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
[21] ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
[41] ok ok ok ok ok ok ok ok ok ok 1  ok ok ok ok ok ok ok ok
[61] ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
[81] ok ok ok ok ok ok ok
Levels: ok ? 1
```

Compositions marked with `ok` have not been classified as outliers. Proven outliers are marked with `1` if there exists one component, such that the datum is not any more extreme in the subcomposition without that component. Finally, data are marked with `?` if no such component exists. Our single outlier can evidently be explained by an error in a single component. It is also possible that a datum becomes regular after removal of several parts: in this case, we may rank the parts by computing which component makes the composition "most" normal (i.e., smallest robust Mahalanobis distance in the subcompositions) after removing it:

```
> OutlierClassifier1(x,type="best")

 [1] ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
[21] ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
[41] ok ok ok ok ok ok ok ok ok ok Cr ok ok ok ok ok ok ok ok
[61] ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok ok
[81] ok ok ok ok ok ok ok
Levels: ok ? Cr Zn Pb
```

Our single detected outlier can thus be explained best by an error in the Cr value. Though maybe, other components could be removed instead of Cr and make the observations regular. If we want a report of all possible components that can be removed to make the data regular, the following command may be useful. It encodes in a binary number all components that can be removed to explain the outlier:

```
> OutlierClassifier1(x,type="all")

 [1] ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok
[16] ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok
[31] ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok
[46] ok  ok  ok  ok  ok  100 ok  ok  ok  ok  ok  ok  ok  ok  ok
[61] ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok
[76] ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok  ok
Levels: ok 100
```

Each datum gets a binary digit of length D (the number of parts), where the ith digit is 1 if removing component i makes the datum regular. Of course, if this does not happen, the ith digit is 0. These tests are based on the χ^2 outlier criterion. For instance, in a $\{A, B, C, D, E\}$-composition, a datum with classifier `00000` may be considered regular (or not explained by any single part), whereas a classifier `01001` means that we may either remove B or E and convert that outlier into a regular observation.

In the example above, only one component can explain the outlier. We can thus say that the outlier could be a single component outlier, and in this case, Cr is the source of the problem.

7.5 Working with Outliers

7.5.1 Random Outliers

Even if the basic distributional assumption (e.g., additive logistic normality) is adequate, it is possible that a value falls pretty far away from the main body of the dataset. In particular, ALN-distributed data spread on the whole simplex; thus, any automatic or semiautomatic outlier detection technique will sooner or later identify a "good" value as an outlier. With a high probability, random outliers should show up phenomenologically as near outliers.

Regarding estimation (e.g., of compositional center or variation matrix), one may be tempted to believe that removing such outliers or using robust statistical methods increases the performance of statistical estimators. However, from mathematical statistics, we know that the classical estimators of mean and variance in an additive lognormal model are the best unbiased estimators. From the perspective of estimation, we would thus use all data, regular and random outliers alike.

This does not imply that the extremal observation still carries contextual information worth being explored: is anything else special on that individual that might suggest further attention? where does the sample come from? On the other hand, we should keep in mind that such "anything special" might be the effect of pure chance. Without further evidence, it will be safer to consider the value as an outlier.

In likelihood-based modeling (ML and Bayesian statistics), random outliers do not need special treatment. They are strange but possible occurrences of the model, and their likelihood is computed in a standard way.

7.5.2 Failed Measurement or Transmission in Individual Components

A technical problem during measurement, a confusion of results, or a transcription error can lead to a wrong value in a single component. Such a problem should show up as a single component error. The typical case is the accidental pollution of a

sample by a pure component in the lab or the confusion of dot and colon as symbols for thousands or floating point when passing files between computers.

In all these cases, the value reported is not informing of the true value, and the error is independent of this actual value. The value can thus be treated like a missing completely at random, if measuring it again or tracing back the error is not possible.

A special case is outliers that are generated by the high relative error near the detection limit. This situation is analog to a MNAR missing value, but (unlike for missing values) here the essential source of problem is known. Even with a constant detection limit, such problem could arise, for instance, (a) for samples substantially smaller than the usual samples or (b) for sample in which one portion is expected to be extremely small according to what the other components tell.

In the first situation (a), all components might have a higher measurement error and even several components might be affected by the problem, in such a way that the observations might not be identified as single component outliers. The information actually important is given by the absolute values or the detection limits scaled to the composition, not by the composition itself. In this case, a simple downweighting of this sample according to its higher measurement variance might be a good solution.

In the second case (b), we could expect the whole sample to be extreme and thus have a strong leverage, i.e., well dominate the parameter estimation. So, on one hand, keeping the sample might distort the estimated parameters, but leaving out the sample could eliminate an important extremal information. If nonstatistical information allows us to conclude that this extremal case can be ignored for the question under consideration, we would leave out the sample or the value by declaring it BDL with a higher detection limit and would try to guess a non-influential replacement value in an EM-algorithm. But if we conclude that the quantitative information is relevant, there is no more work-around than to measure that component again with a method with higher precision and/or lower detection limit.

A Bayesian modeling of wrong observation would need a model for their distribution. Such models can, e.g., be found in Fung and Bacon-Shone (1993) or Madigan et al. (1996). We would need to apply such model for each component individually and integrate over the possible values of the closing constant.

7.5.3 Subpopulations

If the sampled population consists of two or more different subpopulations with different means or variance structures, a single additive lognormal model will fail to capture the distributional shape. If we are lucky and the subpopulations are sufficiently different, they might generate observable secondary modes and thus be detectable as separate groups of outliers. The origin of such problems could be the presence of qualitatively different subpopulations or a systematic grouping in the measurement process, by, e.g., making the same error repeatedly.

While the second problem corresponds to major issues of the processing, the first is an imperfection of the model itself. It might hint to a missing categorical regressor, indexing the several subpopulations. If this categorical variable is observed, a better model might remove the problem. If no explanation for the grouping is observed, nonparametric clustering models or mixing models might be a better choice for the situation. In any case, this type of outliers calls for a scientific understanding of its origin rather than for a straightforward solution.

7.5.4 Polluted Samples

Another origin of special observations is polluted samples. In this case, one may see a mixing effect in some of the actually observed components. If only a single component is affected, we probably see this problem as a single component outlier. If multiple components are affected, the problem is more complicated.

The reported composition of a polluted sample is a mixing of the original composition with a pollutant composition. Mixing is a nonlinear operation in the simplex with respect to the Aitchison geometry. Pollution might affect several samples. If this is the case, it is still improbable that different samples are polluted with the same relative amount of the pollutant. It is thus easy to see a fan of polluted samples along a curve in the Aitchison space that would look like a band around a straight segment in the classical geometry. If the effect of the pollution is strong enough, it could show up as a directional cluster of outliers pointing into the direction of the composition of the pollutant.

For the treatment of polluted samples, we have to take an essential modeling decision. Are we interested in describing the distribution of the unpolluted population, or do we consider the pollution itself also?

In the first case, we want to remove the pollution influence. We may thus assume a dataset with errors, which might be detectable in some samples, but probably not in all of them (since some might be less polluted). Robust statistics might be the best solution to deal with that problem, because they are stable in the presence of errors. However, we should remember that the actual portion of wrong observations is probably larger than the portion of outliers reported by the methods.

In the second situation, a simple (e.g., additive logistic normal) model is insufficient to describe the reality we are interested in. A robust estimation would ignore a major aspect of reality: pollution. However, since the distribution may still be unimodal, maybe skew distributions become useful. Thus, in a first step, a classical lognormal modeling using non-robust estimators might answer rough questions on the dataset reasonably well. The approximate distribution of the pollutant might be guessed from the direction of the effect, and we could try to find the locations of the polluted samples in the field to track down the origin and type of the pollution. However, a better approach should include mixture models.

If this situation corresponds to a single component outlier, it should not be treated as a missing like in the situation of a measurement error.

In a Bayesian approach, pollution could be modeled as convex combinations of random members of a base population and a pollutant population, where the convex coordinate is (with high probability) exactly 1 for the base population and somehow distributed on [0, 1] for the polluted samples. Hyper-parameters for all the distribution of the coordinate and the pollutant population seem necessary, which makes the model quite complicated and not very practical.

7.5.5 Heavy-Tailed Compositional Distributions

The distribution of a compositional population is not necessarily well described by an additive lognormal distribution, like not all univariate values are normal or lognormal. Much of usual "non-normality" can be explained by (approximate) "log-normality" of positive data or "log-ratio-normality" of proportions, but still not everything is well described by additive-logistic-normal distributions. Normal models (including lognormal and additive logistic normal) obviously play a reference role due to the central limit theorem, but not every process fits in the hypotheses of this theorem.

As the name says, the central limit theorem (for compositions) ensures that the distribution of a consecutive random perturbation asymptotically approaches additive logistic normality, especially around the center of the distribution. That means that "interesting" departures from (transformed) normality occur particularly on the tails of the distribution.

Phenomenologically, heavier tails than those corresponding to an additive-logistic-normal model show up as wildly distributed outliers, of which some might be explainable as single component outliers and others not.

Obviously these outliers are not "wrong" or "errors": using the normal model is the wrong thing. Thus, these "outliers" should not be omitted, as it would happen by marking them as missings or treating the dataset with a robust estimator. Instead, the distributional model must be changed.

Some modeling has been tried with skew-normal distributions (Mateu-Figueras and Pawlowsky-Glahn, 2007). Other options could be based on any multivariate heavy-tailed distribution on the ilr plane (e.g., a generalization of the Cauchy distribution, or a multivariate Student-Siegel) or by the non-normally dominated branch of the Aitchison distribution.

In likelihood-based approaches (maximum likelihood or Bayesian), detecting a heavy tail simply means that the distributional model must be one with heavy tails. The problem may be to find a good and flexible model, suitable for (quasi-)analytical treatment.

References

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. London: Chapman & Hall (Reprinted in 2003 with additional material by The Blackburn Press), 416 pp.

Filzmoser, P., & Hron, K. (2008). Outlier detection for compositional data using robust methods. *Mathematical Geosciences, 40*(3), 233–248.

Fry, J. M., Fry, T. R. L., & McLaren, K. R. (2000). Compositional data analysis and zeros in micro data. *Applied Economics, 32*(8), 953–959.

Fung, W. -K., & Bacon-Shone, J. (1993). Quasi-Bayesian modelling of multivariate outliers. *Computational Statistics & Data Analysis, 16*(3), 271–278.

Madigan, D., Raftery, A. E., Volinksy, C. T., & Hoeting, J. A. (1996). *Bayesian model averaging*. Technical report. Colorado State University.

Martín-Fernández, J. A., Barceló-Vidal, C., & Pawlowsky-Glahn, V. (2000). Zero replacement in compositional data sets. In H. Kiers, J. Rasson, P. Groenen, & M. Shader (Eds.), *Studies in classification, data analysis, and knowledge organization (Proceedings of the 7th conference of the International Federation of Classification Societies (IFCS'2000)*, University of Namur, Namur, 11–14 July (pp. 155–160). Berlin: Springer, 428 pp.

Martín-Fernández, J. A., Barceló-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.

Mateu-Figueras, G., & Pawlowsky-Glahn, V. (2007). The skew-normal distribution on the simplex. *Communications in Statistics—Theory and Methods, 39*(6), 1787–1802.

Palarea-Albaladejo, J., & Martìn-Fernández, J. (2008). A modified em alr-algorithm for replacing rounded zeros in compositional data sets. *Computers & Geosciences, 34*, 2233–2251.

Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Maechler, M., et al. (2007). *robustbase: Basic robust statistics*. R package version 0.2-8.

Rousseeuw, P., & Leroy, A. (2003). *Robust regression and outlier detection*. Wiley series in probability and statistics. New York: Wiley, 239 pp.

Tjelmeland, H., & Lund, K. V. (2003). Bayesian modelling of spatial compositional data. *Journal of Applied Statistics, 30*(1), 87–100.

van den Boogaart, K. G., Tolosana-Delgado, R., & Bren, M. (2006). Concepts for handling zeroes and missing values in compositional data. In E. Pirard, A. Dassargues, & H. B. Havenith (Eds.), *Proceedings of IAMG'06—The XI annual conference of the International Association for Mathematical Geology*, University of Liège, Belgium, CD-ROM.

Index

K.G. van den Boogaart and R. Tolosana-Delgado, *Analyzing Compositional Data with R*, Use R!, DOI 10.1007/978-3-642-36809-7, © Springer-Verlag Berlin Heidelberg 2013