Use R!

Raimon Tolosana-Delgado
Ute Mueller

Geostatistics for Compositional Data with R

Springer

Use R!

Use R!

This series of inexpensive and focused books on R is aimed at practitioners. Books can discuss the use of R in a particular subject area (e.g., epidemiology, econometrics, psychometrics) or as it relates to statistical topics (e.g., missing data, longitudinal data). In most cases, books combine LaTeX and R so that the code for figures and tables can be put on a website. Authors should assume a background as supplied by Dalgaard's Introductory Statistics with R or other introductory books so that each book does not repeat basic material.

More information about this series at http://www.springer.com/series/6991

Raimon Tolosana-Delgado • Ute Mueller

Geostatistics for Compositional Data with R

Springer

Raimon Tolosana-Delgado
Helmholtz-Zentrum Dresden-Rossendorf
Helmholtz Institute Freiberg for Resource Technology
Freiberg, Germany

Ute Mueller
School of Science
Edith Cowan University
Joondalup, WA, Australia

ISSN 2197-5736 ISSN 2197-5744 (electronic)
Use R!
ISBN 978-3-030-82567-6 ISBN 978-3-030-82568-3 (eBook)
https://doi.org/10.1007/978-3-030-82568-3

Preface

The need for a specialized geostatistical treatment of compositional data became evident for the first time at the beginning of the 1980s during the doctoral studies of Vera Pawlowsky-Glahn, Raimon's former PhD supervisor. Although she already proposed the fundamental ideas and most of the tools for this task, the matter slumbered for almost two decades out of sight of practitioners and the general geostatistics community. Reasons were some theoretical concerns about the unbiasedness of the proposed estimators, as well as a complete lack of user-friendly software. The advent of **R** in the 2000s changed this: with the appearance of packages "`gstat`" for multivariate geostatistics and "`compositions`" for compositional data analysis, it became possible to mix and merge tools from them and successfully tackle compositional geostatistics with just a few lines of code.

With that idea, we gave a short course on compositional geostatistics in 2012 at Edith Cowan University in Perth and then another during the 2015 Annual Conference of the International Association for Mathematical Geosciences, our common scientific home of many years. For the course at IAMG2015, we wrote a preliminary version of these materials, including methods and code that were state of the art at that time. With that wisdom, and new results with our co-authors in Chaps. 8 and 10, we ended up writing this book and programming "`gmGeostats`", an own new package seamlessly bridging between "`compositions`" and "`gstat`".

The scope of this book is thus the spatial analysis of regionalised compositions, using these three packages within **R**. We cover both compositional data analysis and geostatistics from scratch, so that people new to one or both of these fields can learn them, including basic theory, necessary tools and practical tips. Some knowledge of linear algebra, probability theory, statistics and a minimal R programming help in following the explanations, although they are not absolutely necessary. The book is thus appropriate for a wide variety of readers, from final-year undergraduate students to senior researchers, and from industry practitioners to academics. Our experience, datasets, and examples come all from the geosciences, in particular mining and environmental geochemistry. However, readers should not think that the methods and tools of this book are not relevant to them: compositional geostatistics has already proven relevant in biosciences, ecology, geophysics, geohealth studies,

forensics, pedology and in every scientific field where data in proportions or percentages are geolocalized and modelling the spatial dependence between them is relevant for the scientific question at hand.

The book can be divided into four main blocks, though these are not formally represented in the book structure. The first two chapters (block one) set the framework and provide some background on compositional data analysis. Chapters 3 and 4 (block two) provide compositional exploratory tools for both non-spatial and spatial aspects. Chapters 5 to 9 (block three) cover all necessary aspects of multivariate Gaussian geostatistics for compositional data, including variogram modelling, cokriging, validation, transformations to multivariate normality and cosimulation algorithms. Finally, Chap. 10 (block four) presents the key ideas for a multipoint simulation of compositional data, illustrated in the case of the direct sampling algorithm. Chapter 11 and the appendix are relevant for both the Gaussian and the multipoint routes and can therefore be viewed as endpoints for both blocks three and four. Readers are recommended to get familiar with the materials from the first two blocks before going into the other chapters of the book.

Freiberg (DE), Germany — Raimon Tolosana-Delgado
Perth (AU), WA, Australia — Ute Mueller
July 2020

Acknowledgements

This book would have not been possible without the years of discussion and collaboration with a lot of people, most significantly Gerald van den Boogaart and Hassan Talebi, co-authors of two of the chapters, and Vera Pawlowsky-Glahn, Juanjo Egozcue and Jennifer McKinley, with whom we have collaborated over years on matters of geostatistics and compositional data analysis. Hassan Talebi and Will Patton agreed to serve as guinea pigs and thoroughly checked a preliminary version of this manuscript, greatly improving it. The accompanying package to this book, "`gmGeostats`", also profited from code by Hassan and Gerald, as well as from discussions on the didactics and structure of multivariate geostatistics with many of our Ph.D. students and participants of postgraduate courses past and present.

The International Association for Mathematical Geosciences (IAMG), the German Academic Exchange Service (DAAD) and Universities Australia, as well as our institutions, the Helmholtz-Zentrum Dresden-Rossendorf and Edith Cowan University, provided the framework for our collaboration, covering short and long stays visiting each other; in particular, Edith Cowan University funded a semester of study leave during which Ute visited Raimon in Freiberg.

Last but not least, we thank our partners Dirk and Andreas, who supported us and endured us and our absences during these years of programming and writing together.

Contents

List of Symbols

A	Accuracy of a spatial model
alr	Additive log-ratio transform
$\mathscr{C}[\mathbf{z}]$	Closure of the compositional vector $\mathbf{z}$
clr	Centred log-ratio transform
$d_A(\mathbf{z}, \mathbf{z}')$	Aitchison distance between $\mathbf{z}$ and $\mathbf{z}'$
$d_{AM}(\mathbf{z}, \mathbf{z}'\|\mathbf{S})$	Aitchison-Mahalanobis distance
$d_{DSA}(\mathscr{E}(\mathbf{x}), \mathscr{E}^{TI}(\mathbf{x}^{TI}))$	Distance between data event and data event in training image
$d_H(\mathbf{z}, \mathbf{z}')$	Discrete Hellinger distance between $\mathbf{z}$ and $\mathbf{z}'$
$d_{mm}(\mathbf{z}, \mathbf{z}')$	Manhattan distance between $\mathbf{z}$ and $\mathbf{z}'$
$\mathscr{E}(\mathbf{x})$	Data event at $\mathbf{x}$
$\mathscr{E}^{TI}(\mathbf{x}^{TI})$	Data event in training image TI at $\mathbf{x}^{TI}$
$f_{\mathbf{Z}}(\cdot\|\mathbf{g}, \mathbf{S})$	Probability density of a random composition with additive lognormal distribution, centre $\mathbf{g}$ and spread form $\mathbf{S}$
glr	Generic log-ratio transform
$\boldsymbol{\gamma}(\mathbf{h})$	Univariate variogram at lag $\mathbf{h}$
$g(\cdot\|\theta)$	Unitary variogram model function with parameters θ
G	Goodness of a spatial model
$G(c)$	Grade above cutoff c
$\boldsymbol{\Gamma}(\mathbf{h})$	Variogram matrix at lag $\mathbf{h}$
$\hat{\boldsymbol{\Gamma}}(\mathbf{h})$	Experimental variogram matrix at lag $\mathbf{h}$
$\hat{\mathbf{g}}$	Geometric centre
$i(\cdots)$	Binary-valued function, i.e. a function of several arguments returning either 0 or 1
$I_{cond}(\mathbf{x})$	Indicator random function, with value 1 if the condition indicated is true at location $\mathbf{x}$ and 0 otherwise
ilr	Isometric log-ratio transform
$\kappa(\mathbf{h})$	Spatial diagonalization efficiency at lag $\mathbf{h}$
MAF	Minimum maximum autocorrelation factorization
ME	Cross-validation mean error
MSE	Cross-validation mean square error

$MSDR_1$	Affine equivariant mean square deviation ratio
$MSDR_2$	Mean square deviation ratio
P	Precision of a spatial model
PCA	Principal component analysis
$\boldsymbol{\Psi}$	Transformation from composition to coordinate representation
$\boldsymbol{\Phi}$	Transformation from coordinate representation to composition
pwlr	Pairwise log-ratio transform
$\rho(\cdot\|\theta)$	Correlogram with parameters θ
$\boldsymbol{\sigma}^2_{OCK}(\mathbf{x}_0)$	Ordinary cokriging estimation variance at location $\mathbf{x}_0$
$\boldsymbol{\sigma}^2_{SCK}(\mathbf{x}_0)$	Simple cokriging estimation variance at location $\mathbf{x}_0$
$\boldsymbol{\sigma}^2_{UCK}(\mathbf{x}_0)$	Universal cokriging estimation variance at location $\mathbf{x}_0$
$\boldsymbol{\Sigma}_{OCK}(\mathbf{x}_0)$	Ordinary cokriging error covariance matrix at location $\mathbf{x}_0$
$\boldsymbol{\Sigma}_{SCK}(\mathbf{x}_0)$	Simple cokriging error covariance matrix at location $\mathbf{x}_0$
$\boldsymbol{\Sigma}_{UCK}(\mathbf{x}_0)$	Universal cokriging error covariance matrix at location $\mathbf{x}_0$
$\mathbb{S}^D$	D-part simplex
$\hat{\mathbf{S}}$	Variance-covariance matrix of the log-transformed data set $\ln(\mathbf{Z})$
$\hat{\mathbf{T}}$	Variation matrix
$\mathbf{T}(\mathbf{h})$	Variation-variogram
$\hat{\mathbf{T}}(\mathbf{h})$	Experimental variation-variogram
$T(c)$	Tonnage above cutoff c
TI	Training image used in direct sampling
$\tau(\mathbf{h})$	Relative deviation from diagonality at lag $\mathbf{h}$
totvar($\mathbf{x}$)	Global dispersion of simulated composition at $\mathbf{x}$
$\mathbf{x}$	Spatial location
$\mathbf{y}$	Normal score vector
$\mathbf{Y}$	Normal random vector
$\mathbf{z}$	Compositional vector
$\mathbf{Z}$	Random composition
$\zeta(\mathbf{h})$	Absolute deviation from diagonality at lag $\mathbf{h}$
$\boldsymbol{\zeta}^*_{SCK}(\mathbf{x}_0)$	Simple cokriging estimate at location $\mathbf{x}_0$
$\boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0)$	Ordinary cokriging estimate at location $\mathbf{x}_0$
$\boldsymbol{\zeta}^*_{UCK}(\mathbf{x}_0)$	Universal cokriging estimate at location $\mathbf{x}_0$
$\mathbf{z}^*(\mathbf{x})$	E-type mean of simulated composition at $\mathbf{x}$
$\oplus$	Perturbation
$\ominus$	Inverse perturbation
$\odot$	Powering
$\langle , \rangle_A$	Scalar product

List of Figures

List of Tables

Chapter 1
Introduction

Abstract This chapter provides the framework for the contents of this book. This includes a brief introduction to the problem of geospatial analysis of compositional data and approaches to the solution. Additionally, the data sets and the **R** packages used throughout the book are presented.

1.1 What Is Compositional Geostatistics?

Geostatistical modelling plays a key role in the evaluation of spatially distributed data from a variety of Earth and environmental engineering and science backgrounds, most particularly in natural resources (mining, oil, fisheries, water, etc). Examples from a geological perspective include data collected from a geochemical survey, where it is not uncommon that for each sample concentrations of more than 30 elements are analysed, but also size distribution data and, more recently, mineral proportions data. There are also many applications in the environmental domain, where the issue is more commonly the characterisation of contaminants. The analysis and modelling of such data often require taking into account not only spatial aspects, but also the fact that these data are multivariate, have non-negative values that are often constrained to sum to a prespecified total, and inform of the relative importance or abundance of some parts forming a whole. Thus, they are compositional data. In describing such data one therefore speaks of a regionalised composition.

In particular the constant sum constraint has undesirable consequences, such as inducing spurious correlations, an effect that was recognised quite early on (Pearson, 1897; Chayes, 1960). Several disciples speak of spurious correlation: in the context of compositional data, the term refers to the fact that correlation between two parts changes unpredictably depending on what other parts are considered in the composition. In addition, the non-negativity of compositional data needs to be accounted for in statistical modelling and so standard techniques are not directly applicable. This led to the introduction of *compositional data analysis*, which takes these issues into account. One of the key aspects is the mapping of the D-simplex to $(D-1)$-dimensional real space via one of the several log-ratio transforms. This is the

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_1

topic of Chap. 2, particularly Sect. 2.2. Once the transformation has been effected, standard statistical and geostatistical methods can be applied, as long as these techniques are multivariate. This book follows this straightforward approach and shows how to embed these transformations and the relative nature of regionalised compositions in their spatial analysis.

1.2 Why Use a Compositional Approach?

A multivariate observation is compositional if it is formed by several variables that jointly describe the relative weight, importance or influence of a part with respect to a whole. As mentioned before, compositional data are affected by the spurious correlation problem. As a consequence the common interpretation of correlation is no longer valid, that is, correlation does not represent a valid measure of linear association between two components. This carries over to geo-referenced compositions, as their spatial auto- and cross-correlation functions are as spurious as correlation coefficients are for non-regionalised compositions (Pawlowsky, 1984, 1989; Pawlowsky-Glahn et al., 1995).

It is arguable that this spuriousness has no practical relevance, given that the unbiasedness conditions of cokriging (Chap. 6) strongly limit the influence of any variable on the *other* variables. Together with a historical strong focus of the mining industry on one single variable (one value metal, one contaminant, etc.), this explains the understanding that one can apply univariate geostatistics considering each variable separately. But the twenty-first century has brought several trends that challenge this approach. It is now understood that the value of a mine is strongly controlled by the mineral composition and microstructure of both the ores and the waste; that productivity of an oil, gas or water reservoir is a function of the diagenetic processes forming its host rock, and these in turn of its petrographic composition and texture; that environmental remediation needs and strategies depend on the bioavailability of pollutants, that is, on the form in which these are bound to the soil. All these require the compositional characterisation of the asset or its medium and a modelling strategy capturing the relationships between the various components of the system. At the same time, analytical developments are making this wealth of compositional information measurable in cheaper and more accessible ways. Now, analysing each component separately is no longer a reasonable strategy, as with it there is the risk of losing or blurring those very important links and balances between the variables: regularities stemming from our geochemical, geological, ecological or petrophysical understanding of the system. Just as an example, the interpolated components of a composition are not necessarily positive, and as a whole they will not sum to 100%, if they are simply interpolated separately. Positivity can be enforced by analysing log-transformed variables, but then the constant sum constraint gets even less controlled. Total sum inconsistencies can be circumvented by interpolating one variable less (reconstructing this last variable as 100% minus the sum of the interpolated quantities), but then the output

completely depends on the excluded variable, and the negative component problem gets out of control. Geostatistical simulation makes things just worse.

In the end, the reason for these inconsistencies runs deeper: by treating each variable separately a coherent, multivariate object—a composition—is replaced by a list of individual numbers—its components—ignoring their interdependency in doing so. If one component explains the *relative* importance of one part of the whole, how are we going to understand it if we remove the *other* components from the system? If there is an intrinsic relationship between some of the components, how are we going to capture it, if our modelling efforts ignore it? The goal of this book is to present strategies for the geostatistical analysis of regionalised compositions that allow tackling these problems, delivering compositionally coherent geostatistical models (Chaps. 5 and 8), cokriging (and kriging!) methods (Chap. 6) and simulation strategies (Chaps. 9–11).

1.3 Data

Throughout this book the following data sets will be used to illustrate concepts and techniques as well as for exercises. One of the data sets is derived from a mining context, and the other two are from geochemical surveys. All data are available as illustration data in package "`gmGeostats`".

1.3.1 Windarling Iron Ore Data

This data set consists of a single bench of 6 m long blast hole (BH) samples from an iron ore mine located in the central Yilgarn, Western Australia (Ward & Mueller, 2012, 2013; Ward, 2015). The bench consists of five discrete rotated fault imbricates. The longest strike distance within each horst (constrained by the angular tolerance) is 60 meters.

At each sample location seven analytes and LOI measured in weight percent are available along with the lithotype identified during logging, although only the five main elements of interest for iron ore mining (Fe, SiO_2, Al_2O_3, Mn and P) will be examined. Additionally, lithotype information is available, including schist, chert, hematite and goethite ores. Most of the work here will focus on data from the latter two categories.

The data set is accessible via

```
> data("Windarling", package="gmGeostats")
```

Its working copy will be named as *windarling*.

1.3.2 Tellus Horizon: A Soil Data

The Tellus Project (Young & Donald, 2013) was an extensive geological mapping project undertaken in Northern Ireland that concluded in 2007. During the survey nearly 30,000 soil, stream-sediment and stream-water samples were collected for analysis. The multi-element total concentration data presented comprise XRF analyses of 6862 rural soil samples collected at 20 cm depths on a non-aligned grid at one site per 2 km^2. There are concentration data for eleven oxides, Na_2O, MgO, Al_2O_3, SiO_2, P_2O_5, SO_3, K_2O, CaO, TiO_2, MnO and Fe_2O_3 and the elemental concentrations for Ag, As, Ba, Bi, Br, Cd, Ce, Cl, Co, Cr, Cs, Cu, Ga, Ge, Hf, I, In, La, Mo, Nb, Nd, Ni, Pb, Rb, Sb, Sc, Se, Sm, Sn, Sr, Ta, Th, Tl, U, V, W, Y, Yb, Zn, Zr. In addition the sample id and the spatial coordinates are recorded. Censored data were imputed using published detection limits.

The data set needs to be downloaded from the GSNI website[1] and preprocessed. Within "`gmGeostats`" this may be done by the following command:

```
> getTellus(cleanup = TRUE, TI = TRUE)
```

which prepocesses the data by adding a selection variable called Flag to specify a subset that will be used later on. The data set so obtained is called `TellusASoil`. Throughout the computations in the text the data set will be referred to as *tellus*. If the `TI = TRUE` is set a training image is generated, which will be used in Chap. 10.

1.3.3 National Geochemical Survey of Australia

The National Geochemical Survey of Australia (NGSA) project was part of the 5-year Onshore Energy Security Program managed at Geoscience Australia between 2006 and 2011 (Johnson, 2006). The NGSA was initiated to determine the composition of surface regolith at the continental scale. Sampling took place between 2007 and 2009. The selected sampling medium was catchment outlet sediment from floodplains or similar landforms located near the spill points or lowest points of large catchments, which were derived from terrain and hydrological analysis (Lech and Caritat, 2007). At each site, a surface (0 to 10 cm depth) top outlet sediment or TOS sample and a deeper (on average 60 to 80 cm depth) bottom outlet sediment or BOS sample were collected. For both depths two grain size fractions are available in the data set: coarse at < 2mm and fine at $< 75\mu$m. An area in eastern Western Australia and northwestern South Australia could not be sampled due to access restrictions. In total, 1315 TOS and 1315 BOS samples (including 10% field duplicates) were collected from 1186 catchments.

The overall analyses for 60 elements were obtained. For the subset considered here (Grunsky et al., 2017) the major element analyses were obtained via XRF

[1] Visit http://www.bgs.ac.uk/gsni/Tellus/ to download the original data.

(Al_2O_3, CaO, Fe_2O_3 (total), K_2O, MgO, MnO, Na_2O, P_2O_5, SiO_2, TiO_2, Cl, S) and the remaining elements from the total digestion (fusion followed by HF + HNO3 digestion) followed by inductively coupled plasma-mass spectrometry (ICP-MS) analysis (Ag, As, Ba, Be, Bi, Cd, Ce, Co, Cr, Cs, Cu, Dy, Er, Eu, Ga, Gd, Ge, Hf, Ho, La, Lu, Mo, Nb, Nd, Ni, Pb, Pr, Rb, Sb, Sc, Sm, Sn, Sr, Ta, Tb, Th, U, V, W, Y, Yb, Zn and Zr). Censored data are reported as the negative of the detection limit.

All concentrations are reported as the total elements in parts per million (ppm; where 1 ppm = 1 mg/kg). All the above analyses were carried out on four subsamples at each site (TOS c/g, TOS f/g, BOS c/g and BOS f/g, in the set abbreviated to Tc, Tf, Bc and Bf, respectively). The spatial resolution is 1 sample per 5000 km^2.

The surface geochemistry samples were classified according to the major crustal blocks (MCB) of Australia obtained from simplifying the major crustal boundaries of Korsch and Doublier (2015a,b), and the coordinates were converted into East and North using the Lambert Conformal Conic projection of Australia (with standard parallels at 18 and 36°S latitude central meridian at 134°E longitude and earth ellipsoid GRS80).

The full data set (de Caritat & Cooper, 2011) can be retrieved from Geosciences Australia website[2] and the data set used in this book is accessible via

```
> data("NGSAustralia", package="gmGeostats")
```

For purposes of analysis we will be renaming this data set as *australia*.

1.4 Relevant R Packages

This book makes use of several **R** packages, mostly of the first four:

"`compositions`" is the reference package for the statistical analysis of compositional data in **R**. At the centre of a constellation of other packages on this topic: "`robCompositions`" (Templ et al., 2010), "`zCompositions`" (Palarea-Albaladejo & Martin-Fernandez, 2015) and "`easyCODA`" (Greenacre, 2018), respectively, specialised in robustness—see Sect. 3.5, zero-replacement methods and dimension reduction techniques, "`compositions`" (Boogaart et al., 2020) provides a series of classes and transformations to capture the relative information of compositions, and a wealth of methods including some from these other packages plus basic geostatistics tools (see Chap. 5) and plotting facilities for data and models.

"`gstat`" was devised as a port to **R** of a stand-alone software of the same name (Pebesma, 2004), to deal with geostatistical analysis of spatially dependent multivariate data. The package is thought to capture heterotopic sampling situations, and although it is considered to be a multivariate geostatistics package,

[2] https://www.ga.gov.au/about/projects/resources/national-geochemical-survey.

in reality it does not bring many of the truly multivariate techniques needed to deal with compositional data. In spite of this, it is a well-tested and stable package, and many of the tasks presented in this book will be tackled with it.

"`sp`" is a cornerstone of spatial analysis in **R**, as it brings a set of data classes to contain spatial objects and spatial data, closely following the paradigms of data representation in geographic information systems (Pebesma & Bivand, 2005). Both "`gstat`" and "`gmGeostats`" make use of these data classes. Recently, another package has apppeared, "`sf`" (Pebesma, 2018), which is aimed at replacing "`sp`" in the long term. This book does not consider "`sf`".

"`gmGeostats`" is the core package of this book, developed for three goals: (1) to have a unified platform for classical and modern geostatistical analysis; (2) to serve as a bridge between "`compositions`" and "`gstat`"; and (3) to provide truly multivariate, high-dimensional, large scale geostatistical methods. The package was developed in parallel to writing this book, and it will grow in the future.

The following additional packages are also used, but they are either instrumental or used locally for very specific tasks

"`magrittr`" Its sole contribution is to provide piping functionality to **R**. The pipe `%>%` allows writing `fun(X, args)` as `X %>% fun(args)`. This might seem to be a poor contribution, but it actually allows performing very complex nested calculations in such a way that they are both computationally efficient and easy to read.

"`dplyr`" is a data wrangling package, offering a series of functions (all called as verbs) for data manipulations (`select` columns, `filter` rows, `mutate` variables into new ones, etc.). This has the advantage to nicely integrate with the pipe, jointly with certain convenience functionality in variable selection.

"`MVN`" provides a catalogue of commands for testing multivariate normality of data. This is intensively used in Chap. 8, dealing with transformation of multivariate regionalised data to joint normality.

"`FNN`" stands for "fast nearest neighbor", and the package provides highly efficient implementations of such algorithms, like k-nearest neighbours. These are essentially used in Chap. 10, for migrating irregularly sampled data to a regular grid. The direct sampling algorithm presented in that chapter also makes use of this functionality.

References

Boogaart, K. G. v. d., Tolosana-Delgado, R., & Bren, M. (2020). *Compositions: Compositional data analysis*. R package version 2.0-0.

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

de Caritat, P., & Cooper, M. (2011). National Geochemical Survey of Australia: The geochemical atlas of Australia: Dataset. http://dx.doi.org/10.11636/Record.2011.020.

Greenacre, M. (2018). *Compositional data analysis in practice* (120 pp.). Chapman & Hall/CRC Press.

Grunsky, E., de Caritat, P., & Mueller, U. A. (2017). Using surface regolith geochemistry to map the major crustal blocks of the Australian continent. *Gondwana Research, 46*, 227–239.

Johnson, J. (2006). Onshore energy security program underway. AusGeo News 84. http://www.ga.gov.au/ausgeonews/ausgeonews200612/onshore.jsp.

Korsch, R., & Doublier, M. (2015a). Major crustal boundaries of Australia, and their significance in mineral systems targeting. *Ore Geology Reviews, 76*, 211–228.

Korsch, R., & Doublier, M. (2015b). Major crustal boundaries of Australia. Scale 1:2 500 000. Second edition. http://www.ga.gov.au/metadata-gateway/metadata/record/83223.

Lech, M., & Caritat, P. de (2007). Regional geochemical study paves way for national survey – Geochemistry of near-surface regolith points to new resources. AusGeo News 86. http://www.ga.gov.au/ausgeonews/ausgeonews200706/geochemical.jsp.

Palarea-Albaladejo, J., & Martin-Fernandez, J. (2015). zcompositions – R package for multivariate imputation of left-censored data under a compositional approach. *Chemometrics and Intelligent Laboratory Systems, 143*, 85–96.

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

Pawlowsky, V. (1989). Cokriging of regionalised compositions. *Mathematical Geology, 21*(5), 513–521.

Pawlowsky-Glahn, V., Olea, R. A., & Davis, J. C. (1995). Estimation of regionalised compositions: A comparison of three methods. *Mathematical Geology, 27*(1), 105–127.

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.

Pebesma, E. (2004). Multivariable geostatistics in S: the gstat package. *Computers & Geosciences, 30*(7), 683–691.

Pebesma, E. (2018). Simple features for R: Standardized support for spatial vector data. *The R Journal*, *10*(1), 439–446.

Pebesma, E., & Bivand, R. S. (2005). Classes and methods for spatial data in R. *R News, 5*(2), 9–13.

Templ, M., Hron, K., & Filzmoser, P. (2010). *robCompositions: Robust Estimation for Compositional Data. Manual and package, version 1.4.1.* R-project.

Ward, C. (2015). Compositions, logratios and geostatistics: An application to iron ore. M.Sc Thesis, Edith Cowan University.

Ward, C., & Mueller, U. (2012). *Geostatistics Oslo 2012*, Chapter Multivariate Estimation Using Log Ratios: A Worked Alternative (pp. 333–343). Quantitative Geology and Geostatistics. Springer.

Ward, C., & Mueller, U. (2013). Compositions, log ratios and bias - from grade control to resource. In *Iron Ore 2013 Shifting the paradigm*, Carlton, Australia (pp. 313–320). The Australasian Institute of Mining and Metallurgy.

Young, M. E., & Donald, A. W. (2013). A guide to the Tellus data. http://nora.nerc.ac.uk/509171/.

Chapter 2
A Review of Compositional Data Analysis

Abstract This chapter provides the concepts from compositional data analysis required to prepare compositional data for geostatistical treatment. Specifically we define the term closure, its rationale and caveats, and the various ways of escaping from its curse, i.e. the various forms of log-ratio transformation.

2.1 Compositions

2.1.1 The Closure

Classically, a *composition* has been defined as a vector $\mathbf{z} = [z_1; z_2; \ldots; z_D] = [z_j;\ j = 1, 2, \ldots, D]$ whose components $z_i \geq 0$ are non-negative and sum to a constant κ. The set of points of D-dimensional real space $\mathbb{R}^D$ satisfying these conditions is called the *D-part simplex*, denoted by $\mathbb{S}^D$.

A compositional data set is an $N \times D$ matrix $\boldsymbol{Z} = [z_{ij};\ i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D]$ whose variables are non-negative and, for each observation i, they sum to κ. Notice that in this case the compositions are listed in rows, since typically $N >> D$. In geostatistical applications, the subscript i usually is the label representing the location in space and the compositional data set is referred to as a *regionalised compositional data set*. In this context, the constant sum is almost never encountered, although virtually all multivariate regionalised data sets do actually have a compositional character. Most often, we are dealing with geochemical compositions (of rocks, soils, water, plant tissues, ...), particle size fractions (in soil science), species distribution (in ecology), etc. In many of these cases, the constant sum is not obvious because only a subset of all possible components was analysed/reported, a *subcomposition*. When dealing with a *regionalised subcomposition* whose total sum is not constant, the question is whether this variable "total sum" is relevant information that is desired to be kept, or it is considered an artefact.

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_2

If the total sum must be kept, one way to proceed is to include a *filler variable* in the data set. For example, if all variables are expressed in percentages, then

$$z_{i,D+1} = 100 - \sum_{j=1}^{D} z_{ij}. \tag{2.1}$$

This is often the case in mining exploitation data, where the individual elemental concentrations are considered proportional to the absolute mass abundances of each element (and thus directly relate to the richness of the deposit).

If the total sum is deemed an artefact, then the *only possible* way to proceed is to remove it by reclosing the D variables considered to sum to κ. This is achieved by applying the *closure* operation to each row $\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iD}]$ of the data set,

$$\mathscr{C}[\mathbf{z}_i] = \frac{\kappa}{\sum_{j=1}^{D} z_{ij}} \mathbf{z}_i. \tag{2.2}$$

The need to reclose the data set often arises in exploration geochemistry, where samples are diluted (with variable amounts of water, wind-blown dust, organic matter, etc.) that can distort the actual signal. Another, even clearer case occurs with plant tissue samples, where the total sum typically correlates with the age of the plant. Species compositions counted from traps in ecological studies may also belong to this category, e.g. if they relate to the species that *can* fall in the trap, and not to the whole ecosystem.

Actually, this last strategy can be used in any situation where one is only interested in the relationships between a subcomposition of the original parts available. If $\mathbf{z} = [z_1; z_2; \ldots; z_D]$ is a composition of the parts $\{A_1, A_2, \ldots, A_D\}$, a subcomposition of $S < D$ parts $\{A_{i_1}, A_{i_2}, \ldots, A_{i_S}\}$ is the closed composition $\mathbf{z}_S = \mathscr{C}\left[z_{i_1}, z_{i_2}, \ldots z_{i_S}\right]$. Indeed, each composition can be considered to be a subcomposition of a larger composition that would include the original parts, plus some hypothetical other parts.

Irrespective of the way in which the data have been forced to constant sum, the remainder of this document assumes that this has been done, and from this point on D represents the final number of components considered in the composition or subcomposition of interest, if necessary including the filler.

2.1.2 The R-Package Compositions

To account for the particular aspects of compositional data, Boogaart and Tolosana-Delgado (2008) presented an **R**-package, named "`compositions`": The package can be used in two different ways. The first way is to declare a data set as compositional (by means of functions such as `acomp`) and then analyse the data as desired: in this case, the package will perform the analysis appropriately for

this kind of data, respecting the relative information and returning results that are always valid as compositions. The second way is to manually use the functions the package provides, such as log-ratio transformations (see Sect. 2.2), ternary diagrams (command `plot.acomp`), compositional distances (see Sect. 2.3.2) and so on. Readers are referred to Boogaart and Tolosana-Delgado (2013) if they are interested in more detailed information on the architecture of the package or on the alternative ways to deal with compositional data as presented there.

To load a data set, go to your working directory,

```
> setwd("YOUR_WORKING_DIRECTORY")
```

and use any of the available reading functions, such as `read.csv` and `read_csv` (to read files in comma separated value format), `read.table` and `read_delim` (for tab separated files), or even `read_excel` (for MS Excel files, in package "`readxl`"). Functions `read.csv` and `read.table` are available from base **R**, whereas `read_csv` and `read_delim` are provided by package "`readr`". Here we load the Windarling data set, which is contained in package "`gmGeostats`", and show the first three rows with column names:

```
> library("gmGeostats")
> data("Windarling", package="gmGeostats")
> windarling=Windarling

> head(windarling, n=3)

  Hole_id Sample.West Sample.East West East Easting Northing
1       1           1           0    1    0  -213.4    84.49
2       2           0           0    1    0  -232.4    53.51
3       3           1           0    1    0  -235.1    43.87
  Lithotype     Fe       P   SiO2  Al2O3       S      Mn
1  goethite 0.6328 0.00103 0.0324 0.0192 0.00019 0.00083
2  goethite 0.5974 0.00082 0.0545 0.0431 0.00033 0.00069
3  goethite 0.6756 0.00101 0.0058 0.0032 0.00018 0.00147
       CL    LOI
1 0.00047 0.0355
2 0.00059 0.0393
3 0.00071 0.0182
```

Of these variables, `Easting` and `Northing` give the spatial coordinates, and `Fe` to `LOI` (Loss-on-Ignition) the composition,

```
> wind.coords = dplyr::select(windarling, Easting:Northing)
> wind.compo = dplyr::select(windarling, Fe:LOI)
```

On these lines we have selected the variables making use of the function `select` from package "`dplyr`",[1] which enables the selection of some variables by name just specifying the first and the last name, separated by the double colon `:`.

[1] There is an incompatible function called `select` in package "`MASS`". To ensure using the right function in such cases, the syntax `package::command` is useful.

Once the composition is selected, we can compute the total sum and check whether the data are already closed

```
> wind.totsum = rowSums(wind.compo)
> summary(wind.totsum)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.675   0.714   0.719   0.724   0.728   0.867
```

which shows that there is a significant residual to 100%, which can be captured as a further part in the composition

```
> wind.compo$R = 1-wind.totsum
```

if that residual can be interpreted (in this case, e.g. as the sum of the masses of all other chemical elements not considered in the eight original components).

In other cases, where the total sum is not deemed informative, the appropriate step would be to filter that variability by reclosing, as will be shown with another case study. Here we have soil geochemistry data from Australia,

```
> data("NGSAustralia", package="gmGeostats")
> australia=NGSAustralia

> colnames(australia)

 [1] "ORDER"                "SITEID"
 [3] "DATE SAMPLED"         "EAST"
 [5] "NORTH"                "LATITUDE"
 [7] "LONGITUDE"            "STATE"
 [9] "REGION"               "DUPLICATE CODE"
[11] "DUPLICATE SITEID"     "SAMPLEID"
[13] "GRAIN SIZE"           "DEPTH"
[15] "CODE"                 "Ag ICP-MS mg/kg 0.03"
[17] "Al XRF mg/kg 26"      "As ICP-MS mg/kg 0.4"
[19] "Au FA mg/kg 0.001"    "Ba ICP-MS mg/kg 0.5"
[21] "Be ICP-MS mg/kg 1.1"  "Bi ICP-MS mg/kg 0.02"
[23] "Ca XRF mg/kg 14"      "Cd ICP-MS mg/kg 0.1"
[25] "Ce ICP-MS mg/kg 0.03" "Cl  XRF mg/kg 10"
[27] "Co ICP-MS mg/kg 0.1"  "Cr ICP-MS mg/kg 0.5"
[29] "Cs ICP-MS mg/kg 0.1"  "Cu ICP-MS mg/kg 0.2"
[31] "Dy ICP-MS mg/kg 0.1"  "Er ICP-MS mg/kg 0.03"
[33] "Eu ICP-MS mg/kg 0.03" "F ISE mg/kg 20"
[35] "FeT XRF mg/kg 35"     "Ga ICP-MS mg/kg 0.1"
[37] "Gd ICP-MS mg/kg 0.03" "Ge ICP-MS mg/kg 0.04"
[39] "Hf ICP-MS mg/kg 0.04" "Ho ICP-MS mg/kg 0.02"
[41] "K XRF mg/kg 42"       "La ICP-MS mg/kg 0.1"
[43] "Lu ICP-MS mg/kg 0.02" "Mg XRF mg/kg 60"
[45] "Mn XRF mg/kg 39"      "Mo ICP-MS mg/kg 0.3"
[47] "Na XRF mg/kg 74"      "Nb ICP-MS mg/kg 0.03"
[49] "Nd ICP-MS mg/kg 0.1"  "Ni ICP-MS mg/kg 0.5"
[51] "P XRF mg/kg 22"       "Pb ICP-MS mg/kg 0.1"
[53] "Pd FA mg/kg 0.001"    "Pr ICP-MS mg/kg 0.02"
[55] "Pt FA mg/kg 0.0005"   "Rb ICP-MS mg/kg 0.2"
[57] "S XRF mg/kg 10"       "Sb ICP-MS mg/kg 0.4"
[59] "Sc ICP-MS mg/kg 0.3"  "Si XRF mg/kg 47"
```

```
[61] "Sm ICP-MS mg/kg 0.04" "Sn ICP-MS mg/kg 0.2"
[63] "Sr ICP-MS mg/kg 0.2"  "Ta ICP-MS mg/kg 0.02"
[65] "Tb ICP-MS mg/kg 0.02" "Th ICP-MS mg/kg 0.02"
[67] "Ti XRF mg/kg 30"      "U ICP-MS mg/kg 0.02"
[69] "V ICP-MS mg/kg 0.1"   "W ICP-MS mg/kg 0.1"
[71] "Y ICP-MS mg/kg 0.05"  "Yb ICP-MS mg/kg 0.04"
[73] "Zn ICP-MS mg/kg 0.9"  "Zr ICP-MS mg/kg 0.2"
[75] "LOI Calc mg/kg"       "MCB"
```

```
> australia %>% dplyr::select(17:75) %>%
+   abs %>% rowSums %>% summary
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 469923  514070  537005  545373  561673  909094     157
```

In this data set,[2] a chemical composition with 60 parts is reported in mg/kg (equivalent to ppm), so that the sum can at most be one million (10^6 mg/kg). In addition, the column names indicate the analytical method used for the determination of each element and its detection limit. These names, informative as they are, will have to be shortened later on. For the time being, they indicate the potential presence of values below the detection limit, which are recorded in the relevant cells with a negative number, e.g. Arsenic (As) shows 121 negative numbers,

```
> summary(australia[,"As ICP-MS mg/kg 0.4"]<0)
```

```
 As ICP-MS mg/kg 0.4
 Mode :logical
 FALSE:5127
 TRUE :121
 NA's :11
```

which correspond to values below the detection limit of 0.4 mg/kg. The presence of these negative values was the reason why `abs` (absolute values) was applied before computing row sums.

In this case the total sums may be deemed as non-informative, as most of the elements are actually reported. It makes thus sense to use Eq. (2.2), which in "`compositions`" is available with function `clo`,

```
> aus.compo= australia %>% dplyr::select(17:75) %>% clo
> aus.compo%>% abs %>% rowSums %>% summary
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      1       1       1       1       1       1     157
```

[2] Note the use of the pipe `%>%` from package "`magrittr`", where `X %>% fun` is equivalent to `fun(X)`. Piping is a comfortable way of nesting functions in **R**, easy to write and to read: from the data set `australia`, select some variables, do absolute values, compute the row sums and return their summaries.

If we are interested in the subcomposition formed by As, Au and S, then we can select and reclose them

```
> aus.subcompo = aus.compo[, c(2,3,41)] %>% clo
> head(aus.subcompo)

     As ICP-MS mg/kg 0.4 Au FA mg/kg 0.001 S XRF mg/kg 10
[1,]              0.02439          5.807e-06          0.9756
[2,]              0.02299          4.421e-06          0.9770
[3,]              0.08277         -2.237e-05          0.9172
[4,]              0.05745          5.577e-06          0.9425
[5,]              0.02551         -5.102e-06          0.9745
[6,]              0.07718          8.389e-06          0.9228
```

showing that the values below the detection limit are preserved as minus the detection limit, which is reclosed as well. Detection limits are discussed in greater detail in Sect. 2.4.

2.1.3 Problems of Closed Data

Chayes (1960) is often cited as the first author warning of the problems of *spurious correlation* when working with geochemical compositions. In an influential paper, Chayes warned geoscientists that the Pearson correlation coefficient between any two components of a composition was not an intrinsic property of these two components, but that its magnitude and sign were influenced by the remaining components. The effect was that correlations between components could change randomly by simply considering them as members of different subcompositions. This is the so-called *spurious correlation problem*. For instance, the correlation coefficient of `LOI` and `Al2O3` in the Windarling data set is a moderate positive 0.272 if the whole composition is considered, but it flips to a clear negative −0.703 if `Fe` is disregarded, i.e. if only the gangue[3] composition is considered. What shall we say, is Loss-on-Ignition positively or negatively related with Al_2O_3? In fact, the problem is more general, and Pearson (1897) himself framed it in the more general context of calculating correlation coefficients between ratios that share numerators or denominators, as e.g. closed components of a certain composition, which share the denominator occurring in Eq. (2.2). A potentially more disruptive aspect of the spurious effects of the closure is the fact that the closed numbers do depend on which other components are available for that particular observation: if we have different sets of variables measured for different observations, and their relative abundances have been normalised on each one of them, the resulting numbers do relate to different subcompositions, and they are simply not comparable.

The second problem of using the most classical statistical tools with compositional data is the *negative bias*: because of the constant sum constraint, the

[3] *Gangue*: part of a mineral deposit considered as not valuable.

variance–covariance matrix of a compositional data set sums to zero (both by rows and by columns, being symmetric),

```
> colSums(cov(wind.compo))
```

```
        Fe           P        SiO2       Al2O3           S
-1.008e-19  7.205e-22 -9.783e-20  1.863e-20 -3.607e-22
        Mn          CL         LOI           R
-6.869e-21 -1.894e-22 -1.768e-20  1.673e-19
```

Because each one of these values is the sum of a variance (which is positive) and 9 covariances, some of these covariances must be negative, no matter whether these components are really physically incompatible. This hinders the interpretation of correlation coefficients (a lack of correlation is no longer linked to zero Pearson correlation coefficient) and makes the covariance matrix singular, jeopardising the use of many statistical methods, from covariance matrices or principal component analysis (Chayes & Trochimczyk, 1978; Butler, 1979) to cluster analysis (Butler, 1978). And this is not just a curiosity of classical statistics without relevance for geostatistical modelling: indeed, the same happens to variograms and covariance functions (as will be introduced in Sect. 4.3), which must sum to zero for any lag (Pawlowsky, 1984). The singularity of the covariance matrix makes the fundamental methods of geostatistical estimation and simulation undefined (Pawlowsky, 1989).

2.1.4 Subcompositional Coherence and Scale Invariance

In order to work with compositional data Aitchison (1986) proposed two principles, subcompositional coherence and scale invariance, which should be satisfied by any meaningful method of compositional analysis.

Subcompositional coherence implies that results obtained analysing a subcomposition cannot contradict those obtained analysing the whole composition. We have already seen that Pearson correlation is not subcompositionally coherent, which is otherwise known as the spurious correlation problem. Later it will be seen that the most conventional distance measure of difference between observations, the Euclidean distance, is not subcompositionally coherent.

Scale invariance proposes that results should be the same for a composition $\mathbf{z}$ and any scaled version $p\mathbf{z}$, such as the composition resulting from a change of units from % to ppm. It can be proven mathematically that this condition actually implies working with ratios of the original parts (Aitchison, 1997).

2.1.5 Alternative Frameworks of Compositions

A modern definition of a composition takes it as a vector $\mathbf{z} = [z_1; z_2; \ldots; z_D] = [z_j;\ j = 1, 2, \ldots, D]$ whose components $z_i \geq 0$ do inform of the relative weight,

abundance or importance of a set of parts. As such, $\mathbf{z}$ and $p\mathbf{z}$ (for a positive $p > 0$) are the same composition, because the relative importance of component i vs. component j in both cases is equal: $z_i/z_j = (pz_i)/(pz_j) = p/p \cdot z_i/z_j = z_i/z_j$. Indeed, within this framework the condition of scale invariance is just the definition of a composition! From a mathematical point of view this definition implies that the compositions $p\mathbf{z}$, for any positive real number $p \in \mathbb{R}_+$, are a so-called *equivalence class*, i.e. a set of objects that are equivalent in all respects. Closing them to $\kappa = 1$ or any other total is just choosing a representative of that class (Aitchison, 1997; Barceló-Vidal, 2000; Barceló-Vidal et al., 2001).

Before the work of Aitchison (1986) some of the attempts to avoid the closure problems were based on assuming the existence of an accessory variable T playing the role of a total. In that way, one could recover the original, non-closed values $T\mathbf{z}$, which could then be analysed free of the sum constraint (but not free of the positivity conditions). A good accessory variable for this case is the density (Aitchison, 1997). This idea is particularly relevant for applications in mining and environmental monitoring and will be used in this book when relevant. Alternatively, Pawlowsky-Glahn et al. (2015) proposed a series of approaches for dealing with the case when the composition itself is assumed to provide information about that total.

2.2 Log-Ratio Transformations

Aitchison (1986) was the first to realise that the problems of compositional data came from the fact that they carry only relative information, i.e. any relevant quantity must be expressible as a ratio of the variables considered. For practical and mathematical reasons, he proposed to work with log-ratios of the variables rather than raw data and defined several alternative transformations of which we will use the *pairwise log-ratio transformation* (*pwlr*), the *additive log-ratio transformation* (*alr*) and the *centred log-ratio transformation* (*clr*).

The set of all possible pairwise log-ratios can be organised in a matrix

$$\begin{bmatrix} 0 & \ln\frac{z_1}{z_2} & \cdots & \ln\frac{z_1}{z_D} \\ \ln\frac{z_2}{z_1} & 0 & \cdots & \ln\frac{z_2}{z_D} \\ \vdots & \vdots & \ddots & \vdots \\ \ln\frac{z_D}{z_1} & \ln\frac{z_D}{z_2} & \cdots & 0 \end{bmatrix},$$

or alternatively in a vector containing the columns of the lower triangle,

$$\mathrm{pwlr}(\mathbf{z}) = \left[\ln\frac{z_2}{z_1}, \ldots, \ln\frac{z_D}{z_1}, \ln\frac{z_3}{z_2}, \ldots, \ln\frac{z_D}{z_2}, \ldots, \ln\frac{z_D}{z_{D-1}}\right]. \tag{2.3}$$

The other two log-ratio transformations are, respectively, defined as

$$\boldsymbol{\zeta} = \text{alr}(\mathbf{z}) = \left[\ln \frac{z_1}{z_D}; \ln \frac{z_2}{z_D}; \ldots; \ln \frac{z_{D-1}}{z_D} \right]$$

and

$$\boldsymbol{\xi} = \text{clr}(\mathbf{z}) = \ln \frac{\mathbf{z}}{g(\mathbf{z})}, \qquad g(\mathbf{z}) = \left(\prod_{j=1}^{D} z_j \right)^{1/D}.$$

In this book the logarithm and the exponential of a vector are always applied component-wise. The last two transforms have as inverses, respectively,

$$\text{alr}^{-1}(\boldsymbol{\zeta}) = \text{agl}(\boldsymbol{\zeta}) = \mathscr{C}\left[\exp(\zeta_1); \exp(\zeta_2); \ldots; \exp(\zeta_{D-1}), 1\right]$$

and

$$\text{clr}^{-1}(\boldsymbol{\xi}) = \mathscr{C}\left[\exp(\boldsymbol{\xi})\right].$$

The inverse of the additive log-ratio transformation is known as the *additive generalised logistic transform* (agl; Aitchison, 1986).

Other transformations exist, among which the *isometric log-ratio transformation* (ilr, Egozcue et al., 2003) is often preferred in non-geostatistical compositional data analysis due to its superior theoretical properties. However, for the geostatistical treatment proposed in this document, the choice of transformations is irrelevant. What is relevant though is that results should *not* depend on which transformation was actually applied. The methods presented in this book satisfy this invariance, that is, they are *affine equivariant*.

In "`compositions`", these transformations are available with commands `clr`, `alr`, `ilr` and `pwlr`, together with their inverses `clrInv`, `alrInv`, `ilrInv` and `pwlrInv`.

```
> head(pwlr(wind.compo), n=2)

    P.Fe SiO2.Fe Al2O3.Fe   S.Fe  Mn.Fe  CL.Fe LOI.Fe    R.Fe
1 -6.421  -2.972   -3.495 -8.111 -6.636 -7.205 -2.881 -0.8240
2 -6.591  -2.394   -2.629 -7.501 -6.764 -6.920 -2.721 -0.8194
  SiO2.P Al2O3.P     S.P    Mn.P    CL.P LOI.P   R.P
1  3.449   2.925 -1.6903 -0.2159 -0.7846  3.54 5.597
2  4.197   3.962 -0.9102 -0.1726 -0.3292  3.87 5.772
  Al2O3.SiO2 S.SiO2 Mn.SiO2 CL.SiO2 LOI.SiO2 R.SiO2 S.Al2O3
1    -0.5232 -5.139  -3.664  -4.233  0.09137  2.148  -4.616
2    -0.2347 -5.107  -4.369  -4.526 -0.32698  1.575  -4.872
  Mn.Al2O3 CL.Al2O3 LOI.Al2O3 R.Al2O3   Mn.S   CL.S LOI.S
1   -3.141   -3.710    0.6146   2.671 1.4744 0.9057  5.23
2   -4.135   -4.291   -0.0923   1.810 0.7376 0.5810  4.78
    R.S   CL.Mn LOI.Mn  R.Mn LOI.CL  R.CL R.LOI
```

```
1 7.287 -0.5687  3.756 5.812  4.325 6.381 2.057
2 6.682 -0.1566  4.042 5.944  4.199 6.101 1.902
```

Note that the alr, clr and ilr transformations can be generally expressed as

$$\boldsymbol{\zeta} = \text{glr}(\mathbf{z}) = \boldsymbol{\Psi} \cdot \ln \mathbf{z}, \tag{2.4}$$

where $\boldsymbol{\Psi}$ is a matrix of D columns whose rows sum to zero (in order to produce log-ratios). If $\boldsymbol{\Phi}$ denotes the generalised inverse of $\boldsymbol{\Psi}$, then the inverse is

$$\mathbf{z} = \text{glr}^{-1}(\boldsymbol{\zeta}) = \mathscr{C}\left[\exp(\boldsymbol{\Phi}^t \cdot \boldsymbol{\zeta})\right].$$

In the case of the alr transformation these matrices have size $(D-1) \times D$ and are

$$\boldsymbol{\Psi} = [\boldsymbol{I}_{D-1}; -\mathbf{1}], \qquad \boldsymbol{\Phi} = [\boldsymbol{I}_{D-1}; \mathbf{0}] - \frac{1}{D}\mathbf{1}_{(D-1,D)}, \qquad \text{(for alr)};$$

in the case of the clr transformation these matrices have size $D \times D$ and are

$$\boldsymbol{\Psi} = \boldsymbol{\Phi} = \boldsymbol{I}_D - \frac{1}{D}\mathbf{1} \cdot \mathbf{1}^t =: \boldsymbol{H}, \text{ (for clr)}; \tag{2.5}$$

finally, in the case of the ilr these matrices satisfy $\boldsymbol{\Psi} = \boldsymbol{\Phi}$, and they can be any matrix of $D-1$ pairwise orthonormal rows, i.e. they have size $(D-1) \times D$.

In "`compositions`", the transpose of the matrix $\boldsymbol{\Psi} = \boldsymbol{\Phi}$ for the calculation of an ilr vector can be obtained with command `ilrBase`, e.g.

```
> round(ilrBase(x=wind.compo), digits=3)

     [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]
1 -0.707 -0.408 -0.289 -0.224 -0.183 -0.154 -0.134 -0.118
2  0.707 -0.408 -0.289 -0.224 -0.183 -0.154 -0.134 -0.118
3  0.000  0.816 -0.289 -0.224 -0.183 -0.154 -0.134 -0.118
4  0.000  0.000  0.866 -0.224 -0.183 -0.154 -0.134 -0.118
5  0.000  0.000  0.000  0.894 -0.183 -0.154 -0.134 -0.118
6  0.000  0.000  0.000  0.000  0.913 -0.154 -0.134 -0.118
7  0.000  0.000  0.000  0.000  0.000  0.926 -0.134 -0.118
8  0.000  0.000  0.000  0.000  0.000  0.000  0.935 -0.118
9  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.943
```

More about working with these transformation matrices can be found in Boogaart and Tolosana-Delgado (2013).

2.3 Compositional Geometry of the Simplex

This section briefly reports a more formal characterisation of compositional data and their sampling space. Further details can be found in other textbooks such as Buccianti et al. (2006), Boogaart and Tolosana-Delgado (2013) or Pawlowsky-Glahn et al. (2015).

2.3.1 The Simplex

In statistical theory the sample space of a variable or set of variables (random variable or random vector) is the set of all possible values these variables can physically attain. The sample space of a composition with D parts is called the D-part *simplex* and is often identified with the subset of D-dimensional real space containing vectors of positive components and constant sum to κ,

$$\mathbb{S}^D = \left\{ \mathbf{z} \in \mathbb{R}^D; \quad z_i \geq 0 \quad \wedge \quad \sum_{i=1}^{D} z_i = \kappa \right\}. \tag{2.6}$$

Often we take $\kappa = 1$, 100% or $1,000,000$ ppm. If $D = 3$, the simplex can be visualised as a ternary diagram (Fig. 2.1), which can be obtained with command `plot.acomp`,

```
> wind.compo %>% dplyr::select(SiO2, Al2O3, LOI) %>%
+   plot.acomp(cex=0.2)
> isoPortionLines(by=0.2, col=8)
```

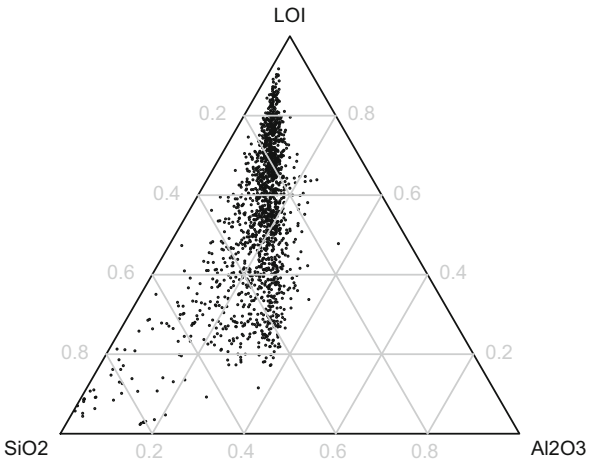

Fig. 2.1 Ternary diagram as representation of the sample space $\mathbb{S}^3$

in this case including some reference lines (with `isoPortionLines`) for ease of reference.

2.3.2 Compositional Distances

Many statistical and machine learning methods require the calculation of a measure of similarity, dissimilarity or distance between observations. The classical Euclidean distance is, in general, inappropriate for compositions, because it is subcompositionally incoherent. For instance, if we compute a few distances between the Windarling compositions with and without iron,

```
> head(dist(clo(wind.compo)))
[1] 0.05032 0.05796 0.03613 0.02145 0.03811 0.02922
> head(dist(clo(wind.compo[,-1])))
[1] 0.12502 0.17564 0.10127 0.05798 0.10794 0.08037
```

the output suggests that the data are more different if iron is not considered, which contradicts the intuition that the differences between observations considering all elements should be *at least as large* as the difference computed without Fe (a concept called *subcompositional dominance*, a particular case of subcompositional coherence). For this reason, Aitchison (1997) introduced a *compositional distance*, based on the pairwise log-ratio,

$$d_A^2(\mathbf{z}, \mathbf{z}') = \frac{1}{D^2} \sum_{i<j}^{D} \left(\ln \frac{z_i}{z_j} - \ln \frac{z'_i}{z'_j} \right)^2 . \tag{2.7}$$

Other distances that have been proposed include the Manhattan distance in closed components (Shurtz, 2003),

$$d_{mm}(\mathbf{z}, \mathbf{z}') = \sum_{i=1}^{D} |z_i - z'_i|,$$

or the discrete Hellinger distance

$$d_H^2(\mathbf{z}, \mathbf{z}') = \frac{1}{2} \sum_{i=1}^{D} \left(\sqrt{z_i} - \sqrt{z'_i} \right)^2 .$$

It should be noted though that, just as the Euclidean distance, all but the Aitchison distance are subcompositionally incoherent.

2.3.3 An Euclidean Vector Space Structure

Billheimer et al. (2001) and Pawlowsky-Glahn and Egozcue (2001) independently showed that the simplex is given an Euclidean space structure with the operations of perturbation $\oplus$ (Aitchison, 1986), powering $\odot$ (Aitchison, 1997) and scalar product $\langle , \rangle_A$, respectively, defined as

$$\mathbf{z} \oplus \mathbf{z}' = \mathscr{C}\left[z_1 z_1'; z_2 z_2', \ldots; z_D z_D'\right], \tag{2.8}$$

$$\lambda \odot \mathbf{z} = \mathscr{C}\left[z_1^\lambda; z_2^\lambda; \ldots; z_D^\lambda\right], \tag{2.9}$$

$$\left\langle \mathbf{z}, \mathbf{z}' \right\rangle_A = \frac{1}{D^2} \sum_{i<j}^{D} \ln \frac{z_i}{z_j} \ln \frac{z_i'}{z_j'}. \tag{2.10}$$

The algebraic geometric properties of these operations make them intuitively analogous to, respectively, vector addition (which is an Abelian group operation), multiplication of a vector by a scalar and dot product of two vectors. Having a scalar product, one can talk of orthogonality of two compositions $\mathbf{z}$ and $\mathbf{z}'$ if $\left\langle \mathbf{z}, \mathbf{z}' \right\rangle_A = 0$, and the distance of Eq. (2.7) corresponds to $d_A^2(\mathbf{z}, \mathbf{z}') = \left\langle \mathbf{z} \ominus \mathbf{z}', \mathbf{z} \ominus \mathbf{z}' \right\rangle_A$, being $\ominus$ the opposite perturbation, $\mathbf{z} \ominus \mathbf{z}' = \mathscr{C}\left[z_1/z_1'; z_2/z_2'; \ldots; z_D/z_D'\right]$.

Given this structure, in the last 20 years the analysis of compositional data has mostly followed what we can call the *principle of working on coordinates*: (1) select a basis of the vector space, (2) express the observations in coordinates with respect to that basis, (3) analyse the coordinates and (4) express your results back in the original units. Interestingly, the clr scores correspond to the coefficients with respect to a generator system of the simplex, the ilr scores are coordinates with respect to an orthonormal basis and the alr scores are coordinates with respect to an oblique basis. The rows of matrix $\boldsymbol{\Phi}$ give the clr coefficients of the $D-1$ vectors of this basis.

Indeed, perturbation and powering can be perfectly represented in terms of sums and products with any of these log-ratio transformations,

$$\mathbf{z} \oplus \mathbf{z}' = \mathrm{glr}^{-1}(\mathrm{glr}(\mathbf{z}) + \mathrm{glr}(\mathbf{z}')),$$
$$\lambda \odot \mathbf{z} = \mathrm{glr}^{-1}(\lambda \mathrm{glr}(\mathbf{z})),$$

but this is not true for scalar products and distances: only the ilr- and clr-transformations satisfy

$$\left\langle \mathbf{z}, \mathbf{z}' \right\rangle_A = \left(\mathrm{clr}(\mathbf{z}), \mathrm{clr}(\mathbf{z}')\right) = \left(\mathrm{ilr}(\mathbf{z}), \mathrm{ilr}(\mathbf{z}')\right), \tag{2.11}$$
$$d_A(\mathbf{z}, \mathbf{z}') = d_{Eucl}(\mathrm{clr}(\mathbf{z}), \mathrm{clr}(\mathbf{z}')) = d_{Eucl}(\mathrm{ilr}(\mathbf{z}), \mathrm{ilr}(\mathbf{z}')).$$

These equivalences are not just trivial, as they allow us to compute the Aitchison distance (Eq. 2.7) in $\mathbf{R}$. If the matrix of interdistances between the compositions

(rows) of a compositional data set is needed, one can just use the built-in function `dist` and evaluate it for a clr-transformed composition,

```
> wind.cdists = dist(clr(wind.compo))
```

2.3.4 *Affine Equivariance and the "Best" Transformation*

With such a wealth of transformations, a typical question practitioners ask is which should be the best one. Indeed, each has its advantages and disadvantages, and choosing one or another is entirely arbitrary. It is thus sensible to consider methods that do not depend on this choice; in particular the methods presented in this book are designed to be *affine equivariant*, to be independent of the actual log-ratio transformation used.

For clarity and ease of use with existing software, both commercial and open source, from this point on we use the alr transformation, in particular with the filler as common denominator when the total sum must be preserved. Wherever the use of the alr may be problematic, this will be explained. Special care must be taken when calculating quantities related to the Euclidean geometry (distances between predictions, orthogonality between residuals and predictions, norm of the residuals, etc.), because of the lack of an equivalent property to Eq. (2.11) for alr scores.

2.4 Zeroes, Missings and Values Below Detection Limit

In the definitions given above and in Eq. (2.6), the value zero was implicitly excluded, in the sense that we did not discuss how to deal with zeroes in the log-ratio transformations. However, zeroes occur quite often in compositional data, sometimes to designate missing values or values below detection limit. There is extensive literature (Aitchison, 1986; Pawlowsky-Glahn et al., 2015; Boogaart & Tolosana-Delgado, 2013) discussing their treatment. The common strategy is to replace these irregular values followed by the application of the spatial techniques described here. The only exception known to the authors is the work by Tjelmeland and Lund (2003) who put forward an integrated approach dealing with zeroes and spatial dependence simultaneously.

If the data set has been encoded using the convention of negative values for values below detection limits, replacement of these missing values can be easily achieved by making use of the function `compositions::zeroreplace`, applied to an `acomp` object as

```
> aus.acomp = australia %>% dplyr::select(17:75) %>% acomp
> aus.acomp.noBDL = zeroreplace(aus.acomp, a=2/3)
```

The output would be a compositional data set where all values below the detection limit have been replaced by 2/3 of those detection limits. This fraction (argument

`a`) can be chosen freely, and some authors prefer other values such as 0.5, or 1 if the variables are chemical components (Boogaart et al., 2011). There are also arguments in favour of using random replacement values, for instance with argument `a=2*runif(sum(is.BDL(aus.acomp)))`, which would replace the missing values with a random uniform fraction between 0 and twice the detection limit.

There are several approaches for incorporating information about the other, non-missing components in the replacement procedure. The idea is always to replace the missing value by some value of the conditional distribution of the missing value given the non-missing ones, be it the expected value or a random realisation. Palarea-Albaladejo and Martín-Fernández (2008) proposed to use an expectation–maximisation algorithm to produce a replacement by conditional expected value, while Martín-Fernández et al. (2003) propose a modification of the same model producing a Markov Chain series of realisations of the missing values.

The problem with all the proposed replacement methods (except Tjelmeland & Lund, 2003) is that they ignore any relevant information about the dependence between the missing value and other collocated variables, or between the missing values at a certain location and its surrounding values. Such replacement strategies (rather *patches*) tend to distort the covariance between variables and create a certain spatial nuggety behaviour. Hence, such unstructured replacements should only be applied if the number of such values below detection limit is small.

Problems

2.1 Subcompositional Coherence and alr Transformation

(a) Compute the alr transformation of the first row of `wind.compo`, and extract the components associated with the parts {Fe, SiO_2, Al_2O_3}.
(b) Build the subcomposition of these three parts plus R (i.e. closing the original values of these four parts to sum to 100%) and compute the alr of this subcomposition. Do you obtain the same values?
(c) Can we thus say that the alr transformation is subcompositionally coherent?

2.2 Subcompositional Coherence and clr Transformation

(a) Compute the clr transformation of the first row of `wind.compo`, and extract the components associated with the parts {Fe, SiO_2, Al_2O_3, R}. Build the subcomposition $\mathbf{z}_S$ of these 4 parts (i.e. closing their 4 raw values to sum to 100%) and compute the clr of this subcomposition. Do you obtain the same values?
(b) Now compute the *differences* between the several clr scores for the same variable, e.g. $\mathrm{clr}_{Fe}(\mathbf{z}) - \mathrm{clr}_{Fe}(\mathbf{z}_S)$. What do you conclude?
(c) Compare differences for pairs of variables within each subset, e.g. $\mathrm{clr}_R(\mathbf{z}) - \mathrm{clr}_{Fe}(\mathbf{z})$ vs. $\mathrm{clr}_R(\mathbf{z}_S) - \mathrm{clr}_{Fe}(\mathbf{z}_S)$. Can we thus say that the clr transformation is subcompositionally coherent?

References

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

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 III Annual Conference of the Int. Association for Mathematical Geology*, Volume I, II and addendum, Barcelona (E) (pp. 3–35, 1100 pp.). International Center for Numerical Methods in Engineering (CIMNE), Barcelona (E).

Barceló-Vidal, C. (2000). Fundamentación matemática del análisis de datos composicionales (77 pp.). Number IMA 00-02-RR.

Barceló-Vidal, C., Martín-Fernández, J. A., & Pawlowsky-Glahn, V. (2001). Mathematical foundations of compositional data analysis. In G. Ross (Ed.), *Proceedings of IAMG'01 – The VII Annual Conference of the Int. Association for Mathematical Geology*, Cancun (Mex), 20 p.

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

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

Boogaart, K. G. v. d., & Tolosana-Delgado, R. (2013). *Analysing compositional data with R* (280 pp.). Heidelberg: Springer.

Boogaart, K. G. v. d., Tolosana-Delgado, R., & Bren, M. (2011). The compositional meaning of a detection limit. In J. J. Egozcue, R. Tolosana-Delgado, M. I. Ortego (Eds.), *Proceedings of the 4th International Workshop on Compositional Data Analysis (2011)*. CIMNE, Barcelona, Spain. ISBN:978-84-87867-76-7.

Buccianti, A., Mateu-Figueras, G., & Pawlowsky-Glahn, V. E. (2006). *Compositional data analysis in the geosciences: From theory to practice*. Special publications (Vol. 264, 212 pp.). London: Geological Society.

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.

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.

Martín-Fernández, J. A., Palarea-Albadalejo, J., & Gómez-García, J. (2003). Markov chain Montecarlo method applied to rounding zeros of compositional data: first approach. In S. Thió-Henestrosa, J. A. Martín-Fernández (Eds.), *Proceedings of CoDaWork'03, The 1st Compositional Data Analysis Workshop*, Girona (E). Universitat de Girona. ISBN:84-8458-111-X. http://ima.udg.edu/Activitats/CoDaWork2003/.

Palarea-Albaladejo, J., & Martín-Fernández, J. A. (2008). A modified EM alr-algorithm for replacing rounded zeros in compositional data sets. *Computers and Geosciences, 34*(8), 2233–2251.

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

Pawlowsky, V. (1989). Cokriging of regionalised compositions. *Mathematical Geology, 21*(5), 513–521.

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., & Lovell, D. (2015). Tools for compositional data with a total. *Statistical Modelling, 15*(2), 175–190.

Pawlowsky-Glahn, V., Egozcue, J. J., & Tolosana-Delgado, R. (2015). *Modeling and analysis of compositional data* (272 pp.). Chichester, UK: John Wiley & Sons.

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.

Tjelmeland, H., & Lund, K. V. (2003). Bayesian modelling of spatial compositional data. *Journal of Applied Statistics, 30*(1), 87–100.

Chapter 3
Exploratory Data Analysis

Abstract In this chapter we will introduce several tools for the exploratory analysis of compositional data, including plots, compositional measures of centre and spread, and compositional principal components to characterise regionalised compositions.

3.1 Graphical Representations

One of the main difficulties of exploring compositional data is the need to capture multivariate relative information, which requires representing several variables at the same time. In this respect individual bivariate scatterplots are quite unsatisfactory. Two different approaches can be used: *ternary diagrams* and parallel plots.

Ternary diagrams (e.g. Figs. 2.1 or 3.1) are a graphical representation of a three-part subcomposition, in the so-called barycentric coordinates. Figure 3.1 shows a ternary diagram with indication of how to read the axes. In "`compositions`", ternary diagrams are available with the generic command `plot.acomp` or alternatively by doing a `plot` of a data set of class "`acomp`". For higher-dimensional subcompositions one could think of using the same idea, for instance to draw a four-part composition within a tetrahedron. This is available with the function `plot3D.acomp`, or alternatively with the function `plot3D` given a 4-component "`acomp`" object (results not shown):

```
> library(dplyr)
> wind.compo %>% select(Fe, SiO2, Al2O3) %>% acomp %>% plot
> wind.compo %>% select(Fe, SiO2, Al2O3, R) %>% acomp %>% plot3D
```

The principle of parallel plotting requires that if a variable is represented in two or more plots, the axes of this variable should be placed in parallel and share the same scaling. In this way, individual observations can be tracked visually through the various plots, as shown in Fig. 3.2. Parallel plotting is the default behaviour of functions such as `pairs`, or `plot` (when applied to "`data.frame`" objects, from standard **R** base). This principle applies to the representation of compositions in any log-ratio transformation as well, for instance with the centred log-ratios (clr)

```
> wind.compo %>% clr %>% pairs
```


R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_3

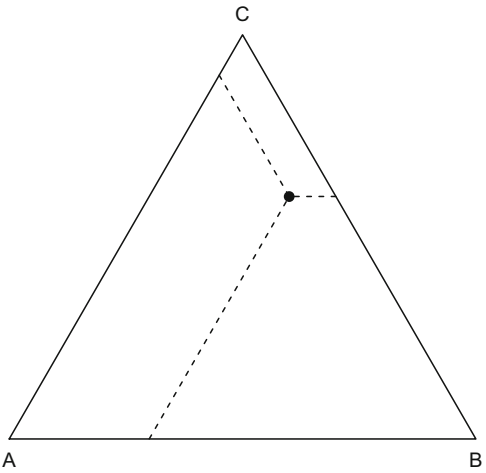

Fig. 3.1 Example of a representation in a ternary diagram, with indication of how to read the proportions: 10% A, 30% B and 60% C

Readers trying this command will observe that the main part of the data collapses on a bunch of undistinguishable dots, and only outliers and extreme values become visible. This is quite a common problem of modern regionalised data sets that often contain several thousands of observations. Scatterplots and ternary diagrams then provide misleading intuitions, as one is invited to think that there is something wrong with the data, having "so many outliers". In these cases, one should resort to representing area densities, such as those provided by bivariate kernel density estimates,

```
> wind.compo %>% clr %>%
+   pairs(panel=vp.kde2dplot, colpalette = spectralcolors)
```

This command produces a plot such as Fig. 3.3, although here only 4 components of the clr-transformed data set are represented, for the sake of saving space.[1] The function `pairs` admits an extra argument `panel`, which can be given a specific function to control the way each bivariate plot is produced. The packages described in this book provide several accessory panel functions, indicated with the suffix `vp.*`, that can be combined with such calls. In the case of Fig. 3.3 the panel function `vp.kde2dplot` creates a bivariate kernel density estimate of the data provided, represents it in a colour scale (red meaning high density, blue meaning low density) and adds some extra indications assessing the linear dependence between the two variables: the regression line of the ordinate variable as a function of the abscissa variable, and the correlation of coefficient between both variables.

When representing compositional data, another general strategy is to utilise the pairwise log-ratio transformation (pwlr) and arrange the results in a matrix of

[1] Note that this is not the same as representing the clr-transformed subcomposition of these four components! See Problem 2.2.

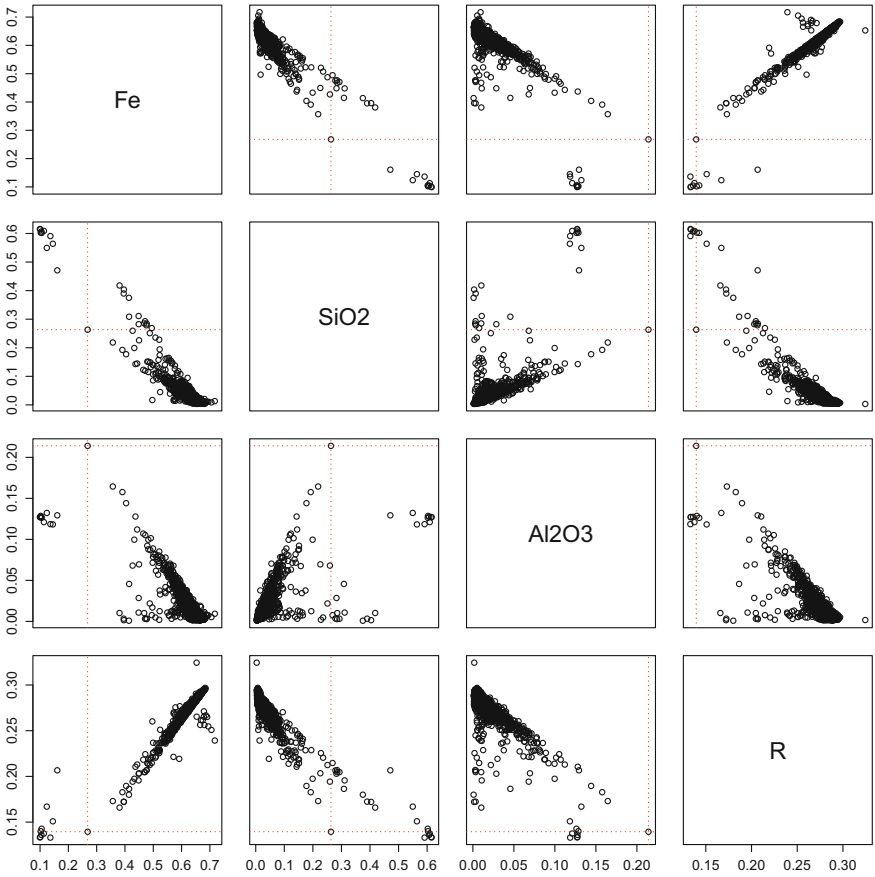

Fig. 3.2 Example of the principle of parallel plotting, and tracking of individual observations, via a matrix of Harker diagrams with the Windarling data set

diagrams, where each plot represents an aspect of the log-ratio. This is the behaviour of function `pairs` when applied to an "`acomp`" object,

```
> wind.compo %>% dplyr::select(Fe, SiO2, Al2O3, R) %>%
+   acomp %>% pairs
```

In this case the output (Fig. 3.4) represents (the default) a kernel density estimate and a histogram of each pairwise log-ratio, using the row variable as numerator and the column variable as denominator. One can control what will be represented by the function `pairs` in each panel, typically by means of the argument `panel`, or even with `lower.panel` and `upper.panel` if one wants different diagrams in the lower and upper triangles of plots. For pairwise log-ratio representations, these panel arguments should be given functions starting with the prefix `vp.lr*`. Advanced users can program their own panel functions after the examples of

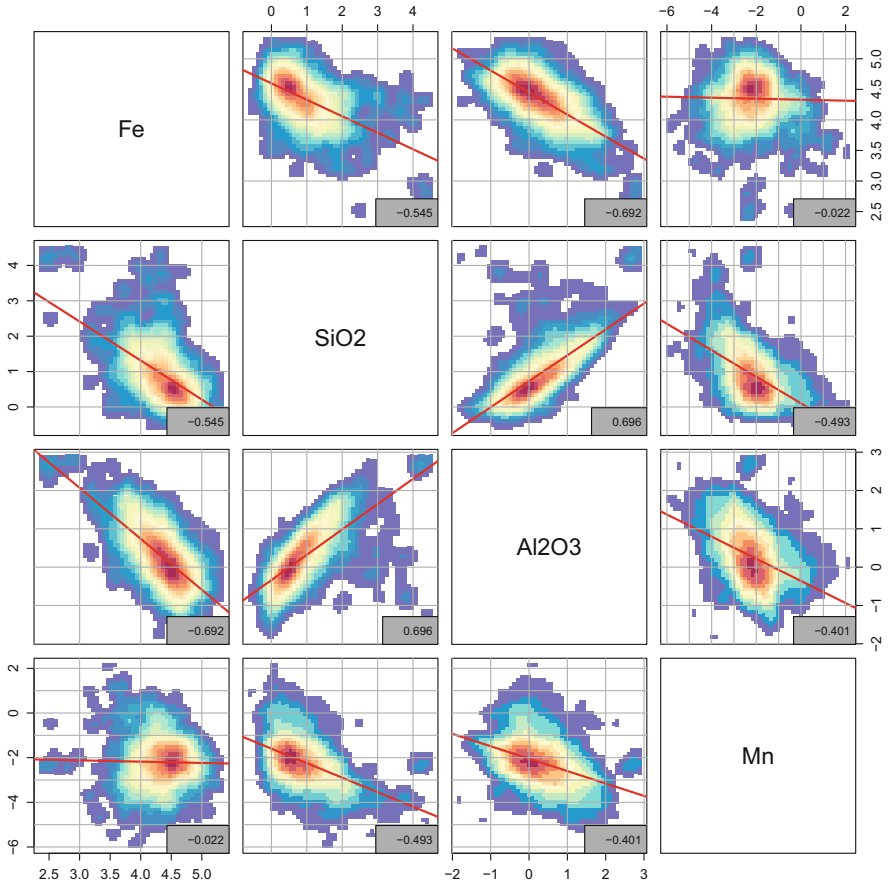

Fig. 3.3 Matrix of bivariate kernel density plots of four components of the clr-transformed Windarling data set

`vp.lrdensityplot` (producing densities and histograms) or `vp.lrboxplot` (producing boxplots). Remember that a boxplot is a compact display of the distribution of the data by showing a box with the three quartiles (i.e. the values leaving 25%, 50% and 75% of the data behind) and whiskers extending along the typical values (with a total length of at most 4 times the length of the box, roughly corresponding to 99% of the data for a normally distributed variable), complemented with dots indicating extreme values or outliers.

This last diagram is actually easier with function `boxplot` applied to an "`acomp`" object

```
> wind.compo %>% dplyr::select(Fe, SiO2, Al2O3, R) %>%
+   acomp %>% boxplot
```

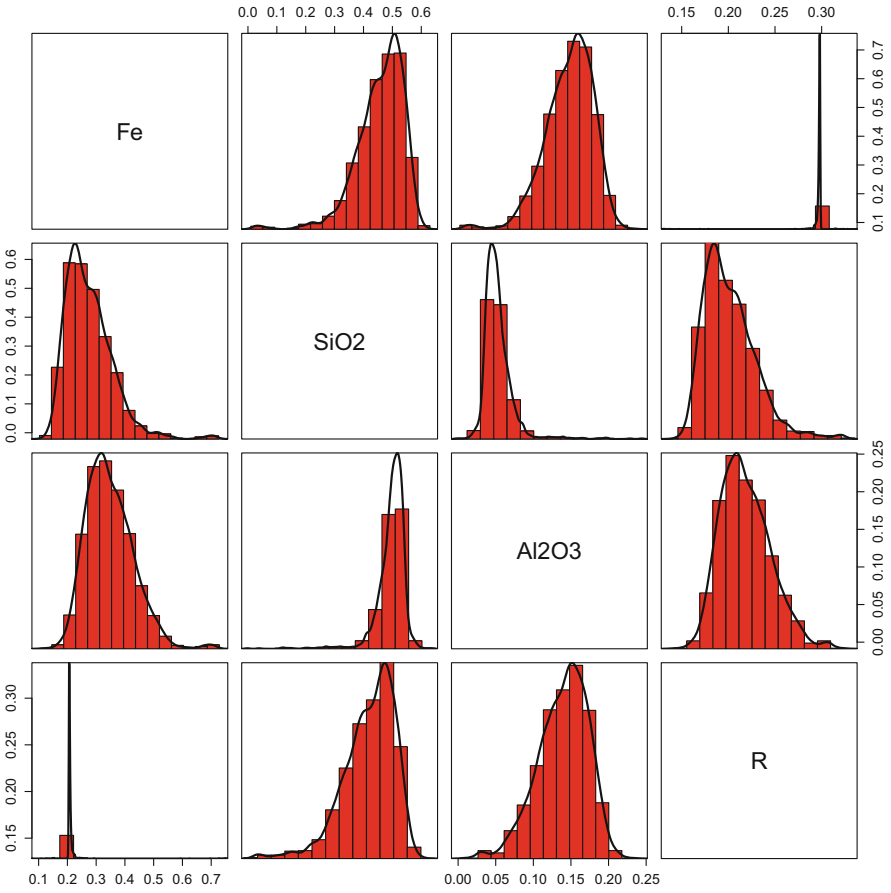

Fig. 3.4 Matrix of pairwise representation of a composition, where each panel represents the estimated density distribution of the component in the row divided by the component in the column

Figure 3.5 shows the output. Note that in this representation all vertical axes span the same orders of magnitude, i.e. their logarithmic length is the same. In this way we can compare the apparent spread of the data and of the boxes across log-ratios.

3.2 First and Second Order Moments

Together with graphical representations, descriptive statistics of the data are also useful, notably central tendency (means) and spread or codependence measures (variances/covariances).

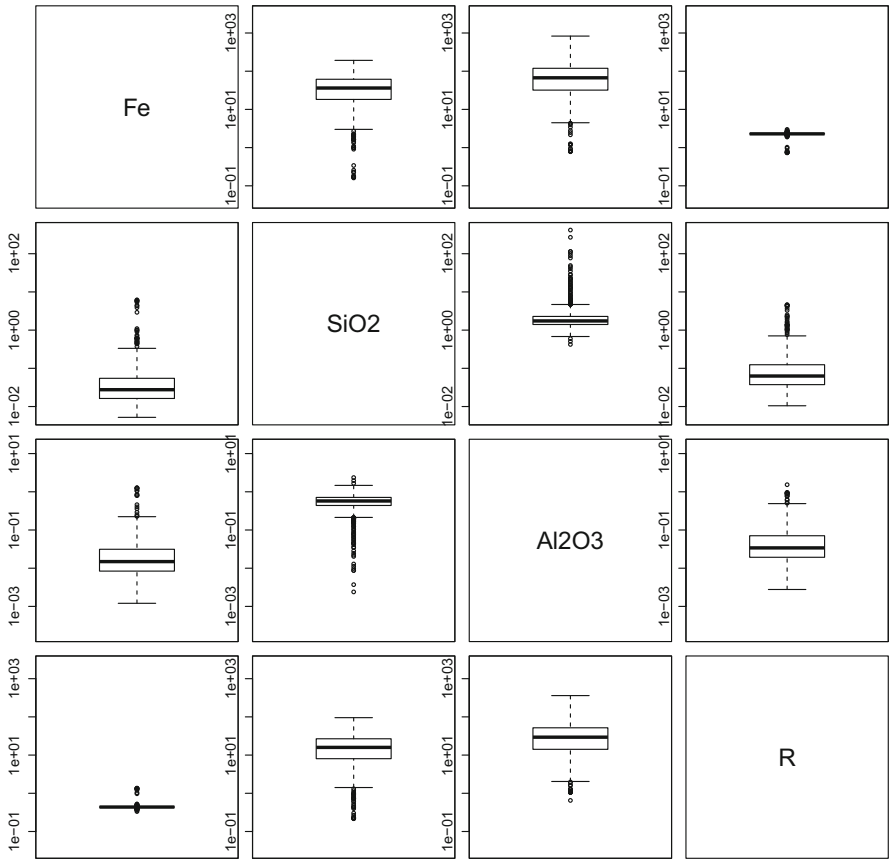

Fig. 3.5 Pairwise log-ratio boxplots of a four-component subcomposition; vertical scales in log scaling

To describe the central tendency of a compositional data set $\mathbf{Z} = [z_{ij}]$, two options are possible: to use the *geometric centre*, or to calculate the mean of a set of log-ratio transformed scores. The geometric centre $\hat{\mathbf{g}}$ is estimated as

$$\hat{\mathbf{g}} = [\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_D] = \mathscr{C}\left[\left(\prod_{i=1}^{N} z_{i1}\right)^{1/N}, \left(\prod_{i=1}^{N} z_{i2}\right)^{1/N}, \ldots, \left(\prod_{i=1}^{N} z_{iD}\right)^{1/N}\right] \quad (3.1)$$

$$= \mathscr{C}\left[\exp\left[\frac{1}{N}\sum_{i=1}^{N} \ln z_{i1}, \frac{1}{N}\sum_{i=1}^{N} \ln z_{i2}, \ldots, \frac{1}{N}\sum_{i=1}^{N} \ln z_{iD}\right]\right],$$

while the log-ratio (alr, clr or ilr) mean vector $\bar{\boldsymbol{\zeta}}$ is

$$\bar{\boldsymbol{\zeta}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\zeta}_i = \frac{1}{N}\sum_{i=1}^{N}\mathrm{glr}(\mathbf{z}_i) = \boldsymbol{\Psi}\cdot\left(\frac{1}{N}\sum_{i=1}^{N}\ln\mathbf{z}_i\right),$$

i.e. the geometric mean can be computed directly from the arithmetic mean of the log-transformed composition.

It is easy to show (Pawlowsky-Glahn et al., 2015) that these two central statistics are closely related, as for any log-ratio transform glr (alr, clr, ilr or pwlr),

$$\bar{\boldsymbol{\zeta}} = \mathrm{glr}(\hat{\mathbf{g}}). \tag{3.2}$$

Within "`compositions`", the geometric centre is obtained directly by using the command `mean` on an "`acomp`" object,

```
> (g = mean(acomp(wind.compo)))
```

```
      Fe         P      SiO2     Al2O3         S        Mn        CL
0.642865 0.001210 0.020998 0.010936 0.000205 0.000922 0.000499
     LOI         R
0.039410 0.282955
attr(,"class")
[1] acomp
```

log-ratio means can be computed with similar nested commands as well

```
> mean(clr(wind.compo))
```

```
    Fe      P   SiO2  Al2O3      S     Mn     CL    LOI      R
 4.349 -1.927  0.927  0.275 -3.700 -2.198 -2.811  1.557  3.528
attr(,"class")
[1] "rmult"
```

or making use of Eq. (3.2), by computing the clr of the geometric mean

```
> clr(g)
```

```
    Fe      P   SiO2  Al2O3      S     Mn     CL    LOI      R
 4.349 -1.927  0.927  0.275 -3.700 -2.198 -2.811  1.557  3.528
attr(,"class")
[1] "rmult"
```

Note that this last computation incorporates a summary of the kinds of missing values and zeroes found in this "data set", even if in this case this only contains one row with the mean. More details about missing values and zero classifications can be found in Sect. 2.4.

To describe the spread and codependence structure of the compositional data set, one must necessarily work with log-ratio transformed data. If the pwlr is used, it suffices to calculate all variances,

$$\hat{t}_{jk} = \frac{1}{N-1} \sum_{i=1}^{N} \left(\ln \frac{z_{ij}}{z_{ik}} - \ln \frac{\hat{g}_j}{\hat{g}_k} \right)^2, \tag{3.3}$$

which can be arranged in the *variation matrix* $\hat{\mathbf{T}} = [\hat{t}_{jk}]_{j,k=1,2,\ldots,D}$. In contrast, using one of the vector-valued log-ratio transformations produces a variance–covariance matrix,

$$\hat{\mathbf{S}} = \frac{1}{N-1} \sum_{i=1}^{N} (\boldsymbol{\zeta}_i - \bar{\boldsymbol{\zeta}}) \cdot (\boldsymbol{\zeta}_i - \bar{\boldsymbol{\zeta}})^t .$$

As happens with the central tendency estimators, the spread estimators are also related to one another, and to the variance–covariance matrix of the log-transformed data set $\ln \mathbf{Z}$, by

$$\hat{\mathbf{S}} = \boldsymbol{\Psi} \cdot \mathrm{var}[\ln \mathbf{Z}] \cdot \boldsymbol{\Psi}^t = -\frac{1}{2} \boldsymbol{\Psi} \cdot \hat{\mathbf{T}} \cdot \boldsymbol{\Psi}^t . \tag{3.4}$$

Derivations of all these expressions can be found e.g. in Boogaart and Tolosana-Delgado (2013). Given these relationships, we consider $\hat{\mathbf{T}}$ and $\hat{\mathbf{S}}$ (calculated using any log-ratio) as equivalent alternative expressions of the same *object*, which we call a spread form.

The default way to compute the spread form of an "`acomp`" object is to produce the variance–covariance matrix of the clr-transformed scores, i.e.

```
> var(acomp(wind.compo))
> var(clr(wind.compo))
```

produce the same results (not shown here to save space). Alternatively, one can use `var(ilr(.))` or `var(alr(.))` to compute the variance–covariance matrix of ilr- resp. alr-transformed scores. Otherwise, the *variation matrix* is available with

```
> variation(acomp(wind.compo))
```

```
            Fe     P  SiO2 Al2O3     S   Mn    CL   LOI       R
Fe     0.00000 0.163 0.953 0.959 0.568 1.07 0.240 0.171 0.00727
P      0.16297 0.000 1.111 1.015 0.658 1.05 0.382 0.209 0.16662
SiO2   0.95282 1.111 0.000 0.310 0.869 2.15 0.692 0.789 0.88454
Al2O3 0.95947 1.015 0.310 0.000 0.662 1.94 0.722 0.779 0.89944
S      0.56803 0.658 0.869 0.662 0.000 1.72 0.455 0.441 0.55546
Mn     1.07240 1.050 2.155 1.943 1.716 0.00 1.334 0.980 1.04795
CL     0.23977 0.382 0.692 0.722 0.455 1.33 0.000 0.280 0.22114
LOI    0.17111 0.209 0.789 0.779 0.441 0.98 0.280 0.000 0.15805
R      0.00727 0.167 0.885 0.899 0.555 1.05 0.221 0.158 0.00000
```

i.e. by using the command `variation` on an "`acomp`" object. Finally, there is a family of functions for converting between these various representations of spread.

3.3 PCA and Biplots

Principal component analysis (PCA) is the cornerstone of multivariate exploratory analysis for quantitative variables. Given a (N-observations, zero-mean, scaled) data set $\mathbf{Y}$ and its covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{Y}^t\mathbf{Y}$, PCA expresses the covariance matrix through the spectral decomposition as

$$\mathbf{S} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^t. \tag{3.5}$$

Here $\boldsymbol{\Lambda}$ is a diagonal matrix with non-negative diagonal elements (the *singular values*) that are arranged in decreasing order, and $\mathbf{Q}$ is an orthogonal matrix whose columns represent the *eigenvectors* or *principal directions* of $\mathbf{S}$. Hence $\mathbf{Q}^t\mathbf{Q} = \mathbf{I}$, as the eigenvectors have norm one and are mutually orthogonal. The matrix $\mathbf{Q}$ represents a generalised rotation of the data space where the covariance matrix becomes a diagonal matrix: this diagonal matrix is $\boldsymbol{\Lambda}$. Thus, the eigenvalues are the variances of the rotated data, and the scores of the rotated data can be obtained as $\mathbf{Y}^* = \mathbf{YQ}$. Further details about this decomposition can be found in Sects. A.2–A.4.

We can obtain a PCA with compositional data by taking $\mathbf{Y}$ as the centred, clr-transformed composition.

```
> pc = wind.compo %>% clr %>% princomp
> names(pc)
```

```
[1] "sdev"     "loadings" "center"   "scale"    "n.obs"
[6] "scores"   "call"
```

This provides a list of outputs, including the eigenvectors (element `loadings`), the standard deviations of the rotated data (element `sdev`, equivalent to the square roots of the eigenvalues) and the scores of the rotated data (element `scores`). Element `center` gives the vector of clr-means that the function `princomp` uses to centre the data set.

These results can be used to create a series of graphical results of interest in exploratory data analysis, see for example (Grunsky & Caritat, 2020; McKinley et al., 2018). The most common are the *scree plot* and the *biplot*. The scree plot, obtained with function `screeplot`, is a bar diagram of the eigenvalues, appropriate for visually evaluating the variance of each principal component. Typically it is used to determine the practical dimensionality of the data set, e.g. by counting how many principal components are needed to reach a fixed proportion of total variance. This can also be plotted with

```
> (propvar=cumsum(pc$sdev^2)/sum(pc$sdev^2))
```

```
Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
 0.494  0.728  0.845  0.905  0.947  0.977  0.999  1.000  1.000
```

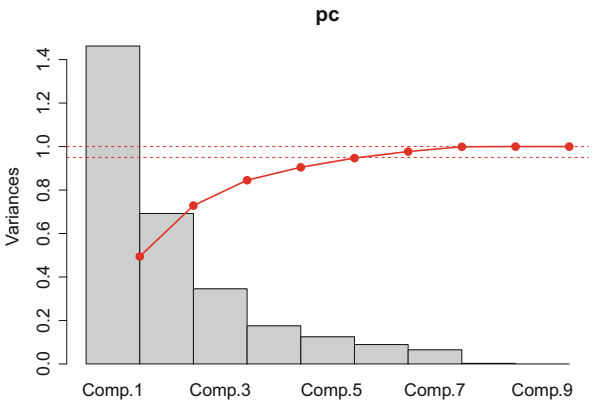

Fig. 3.6 Scree plot of a clr-PCA of the Windarling data, and cumulative proportion of explained variance with 100% and 95% reference lines

```
> screeplot(pc, space=0)
> lines(propvar, lwd=2, col=2, type="o", pch=19)
> abline(h=c(0.95,1), lty=2, col=2)
```

code producing a scree plot and a representation of the cumulative variance explained (Fig. 3.6). In this case, for instance, the first two principal components explain almost 75% of the total variability, and to reach 95% six PCs are required.

Another diagram produced with PCA is a biplot. This is a simultaneous display of both the loadings and the scores in such a way that their spreads are comparable, and the loadings are scaled by the standard deviations of the PCs (Aitchison & Greenacre, 2002). The diagram can be obtained with the **R** function `biplot`, or better for exploratory purposes, with the function `coloredBiplot` from the package "`compositions`"

```
> coloredBiplot(pc, choices = 1:2, xlabs.pc=1, cex=c(0.5,1),
+               col=1, xlabs.col=windarling$Lithotype)
> legend("bottomright", fill=1:4,
+        legend=levels(windarling$Lithotype))
```

Figure 3.7 provides the result of this chunk, where we have controlled which PCs are displayed (the first two, with `choices=1:2`), the symbol and colour for the scores (`xlabs.pc` and `xlabs.col`), the colour of the variable labels (`col`) and the size of both score dots and variable names (`cex`).

A compositional biplot can help in elucidating the compositional dependence structure, provided that the proportion of variance explained by the first two PCs is high enough. If the labels of three or more components fall along a line, these components form a subcomposition strongly dominated by a one-dimensional trend. This is the case for (Mn, Al_2O_3, SiO_2) and (P, Fe, R). The corresponding subcompositions are displayed in Fig. 3.8. Furthermore, if the labels of two components are plotted close to each other, then these two components are practically proportional, as e.g. Fe and Rest, in which case any ternary diagram with these two variables will show all data falling on a line (Fig. 3.8, right). In this case, the figure shows that all data fall on a constant Fe/R line, except for the schist samples, which exhibit

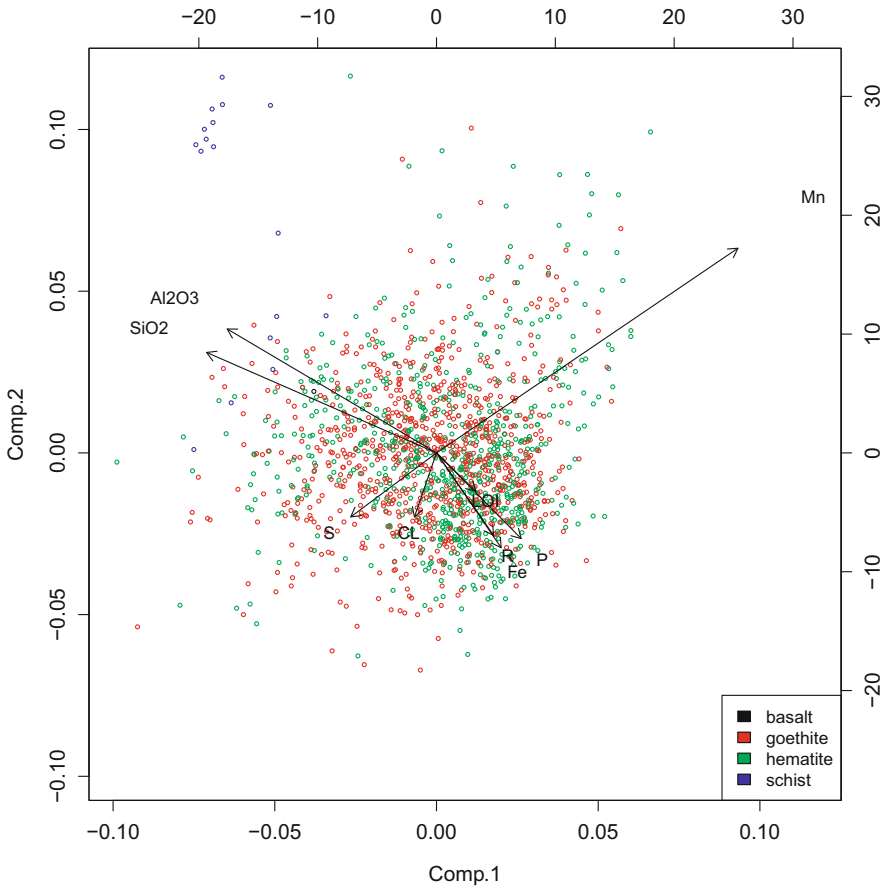

Fig. 3.7 Coloured biplot of the first two PCs of a clr-PCA of Windarling data

a higher Rest than expected for their iron content. A third aspect of biplots that is often useful is the relative location of sample points with respect to the variables: data points falling near a label of a variable typically exhibit values on that (clr-transformed) variable above its mean value. This often helps in understanding why a certain observation is anomalous. For instance in the Windarling case the location of the schist samples in the biplot of Fig. 3.7 suggests that these are richer in SiO_2 and Al_2O_3 than the average of the data, something not at all unexpected. Further details about the interpretation of biplots can be found in Aitchison and Greenacre (2002). For the sake of completeness, the following chunk provides the code used for creating one of the ternary diagrams.

```
> wind.compo %>% dplyr::select(Mn, Al2O3, SiO2) %>% acomp %>%
+   plot(center=T, pca=T, col.pca=1, col=windarling$Lithotype)
```

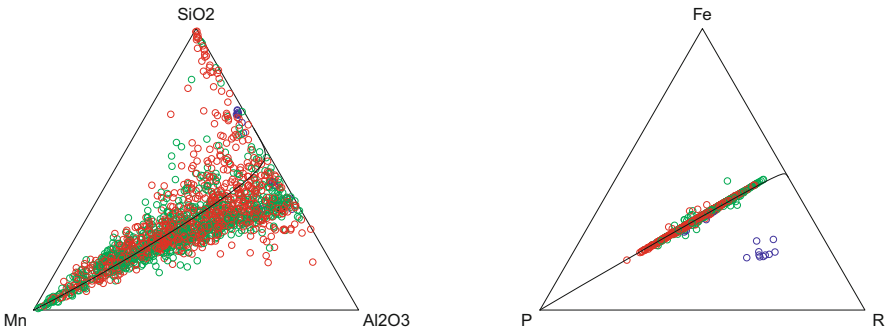

Fig. 3.8 Ternary diagrams of the Windarling data set, with indication of first principal component of the subcomposition. Colour legend after Fig.3.7

3.4 Additive Logistic Normality (ALN) and Mahalanobis Metric

The last concept of classical statistics for compositions required in geostatistics is that of a normal distribution. We introduce it by making use of the *Aitchison–Mahalanobis distance* between two compositions $\mathbf{z}$ and $\mathbf{z}'$ with respect to a spread form $\mathbf{S}$ (w.r.t. to an alr or ilr linked to the matrix $\boldsymbol{\Psi}$), which is defined as

$$\begin{aligned} d^2_{AM}(\mathbf{z}, \mathbf{z}'|\mathbf{S}) &= (\text{glr}(\mathbf{z}) - \text{glr}(\mathbf{z}'))^t \cdot \mathbf{S}^{-1} \cdot (\text{glr}(\mathbf{z}) - \text{glr}(\mathbf{z}')) \\ &= (\ln \mathbf{z} - \ln \mathbf{z}')^t \cdot \boldsymbol{\Psi}^t \cdot \mathbf{S}^{-1} \cdot \boldsymbol{\Psi} \cdot (\ln \mathbf{z} - \ln \mathbf{z}'). \end{aligned} \tag{3.6}$$

The Aitchison–Mahalanobis distance can be computed using any log-ratio transformation as long as the spread form $\mathbf{S}$ is transformed accordingly; the distances computed will not change (Filzmoser & Hron, 2008; Filzmoser et al., 2009), i.e. the Aitchison–Mahalanobis distance is affine equivariant. This distance is a Euclidean metric, that is, it is associated with an inner product,

$$\langle \mathbf{z}, \mathbf{z}'|\mathbf{S}\rangle_{AM} = (\text{glr}(\mathbf{z}))^t \cdot \mathbf{S}^{-1} \cdot (\text{glr}(\mathbf{z}')) = (\ln \mathbf{z})^t \cdot \boldsymbol{\Psi}^t \cdot \mathbf{S}^{-1} \cdot \boldsymbol{\Psi} \cdot \ln \mathbf{z}'.$$

Given that $\mathbf{S}$ is always a positive definite matrix, it admits the decomposition $\mathbf{S} = \mathbf{R} \cdot \mathbf{R}^t$. Matrix $\mathbf{R}$ can be found for instance with the Cholesky decomposition, or through an eigenvalue decomposition of $\mathbf{S}$, as explained in Sect. A.2, as $\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}$. One way or another, this matrix allows to *spherify* the Mahalanobis geometry, i.e. to compute Mahalanobis distances and inner products making use of a conventional Euclidean distance and dot product, respectively. The spherified coefficients of a composition $\mathbf{z}$ are

$$\boldsymbol{\eta}(\mathbf{z}) = \mathbf{R}^{-1} \cdot \text{glr}(\mathbf{z}). \tag{3.7}$$

It is left as an exercise to show that this is indeed a spherification.

A random composition $\mathbf{Z}$ is said to have an *additive logistic normal distribution* (a.k.a. *normal distribution on the simplex*) if its vector of log-ratio transformed scores $\boldsymbol{\zeta}$ has a multivariate normal distribution (Aitchison, 1986; Mateu-Figueras, 2003; Mateu-Figueras & Pawlowsky-Glahn, 2008). That is true for *any* log-ratio transformation (alr, clr, ilr and even pwlr). The probability density function of this random composition can be written (up to the Jacobian of the transformation) as

$$f_{\mathbf{Z}}(\mathbf{z}|\mathbf{g},\mathbf{S}) = \frac{1}{(2\pi\cdot\det(\mathbf{S}))^{(D-1)/2}}\exp\left(-\frac{1}{2}d^2_{AM}(\mathbf{z},\mathbf{g}|\mathbf{S})\right),$$

where $\det(\mathbf{S})$ is the determinant of $\mathbf{S}$.

Although spatial dependence makes such practice dubious or even useless, it is customary in non-spatial exploratory analysis to check for normality of the data, be it visually (through a diagram) or quantitatively (through a goodness-of-fit test). The normality hypothesis for a compositional "`acomp`" object is easy to test with command `acompNormalGOF.test`, checking if the ilr-transformed composition passes an energy test for multivariate normality (Székely & Rizzo, 2005; Rizzo & Székely, 2008). Diagrams for visually checking normality of a composition can be obtained based on the pairwise log-ratio pairs plots, such as Fig. 3.4, where each panel represents a conventional graphical check of normality, for instance a QQ-plot. This is the effect of command `qqnorm` applied to an "`acomp`" object:

```
> wind.compo %>% dplyr::select(Fe, SiO2, Al2O3, R) %>%
+   acomp %>% qqnorm
```

The diagram obtained (Fig. 3.9) shows a reasonable fit for the log-ratio of Al_2O_3 vs. R, and a very bad one for Fe vs. R (with all other log-ratios placed in between). In particular the QQ-plot of Fe vs. R shows the presence of a very small group of perhaps 10 samples radically different from the other samples. This small group of samples may be polluting the representation; they appear to be outliers.

3.5 Outliers and Robustness

The detection (and eventually, filtering) of outliers is one of the tasks of exploratory analyses. But what is an outlier? Conventionally, an outlier is an observation that does not follow the *probability distribution model fitting to the rest of the data*. Unfortunately, there is a certain circular reasoning in that definition because how are we meant to know what that model is without previously knowing which data follow it and which are outliers? There are nevertheless methods for assessing the presence and number of outliers (without necessarily identifying them), and for certain kinds of outliers these can even be detected. Package "`compositions`" contains functions for several of these techniques. The discussion of these methods occupies an entire chapter of the *useR* series book (Boogaart & Tolosana-Delgado, 2013) on this package. What follows is a summary of the most relevant functionalities

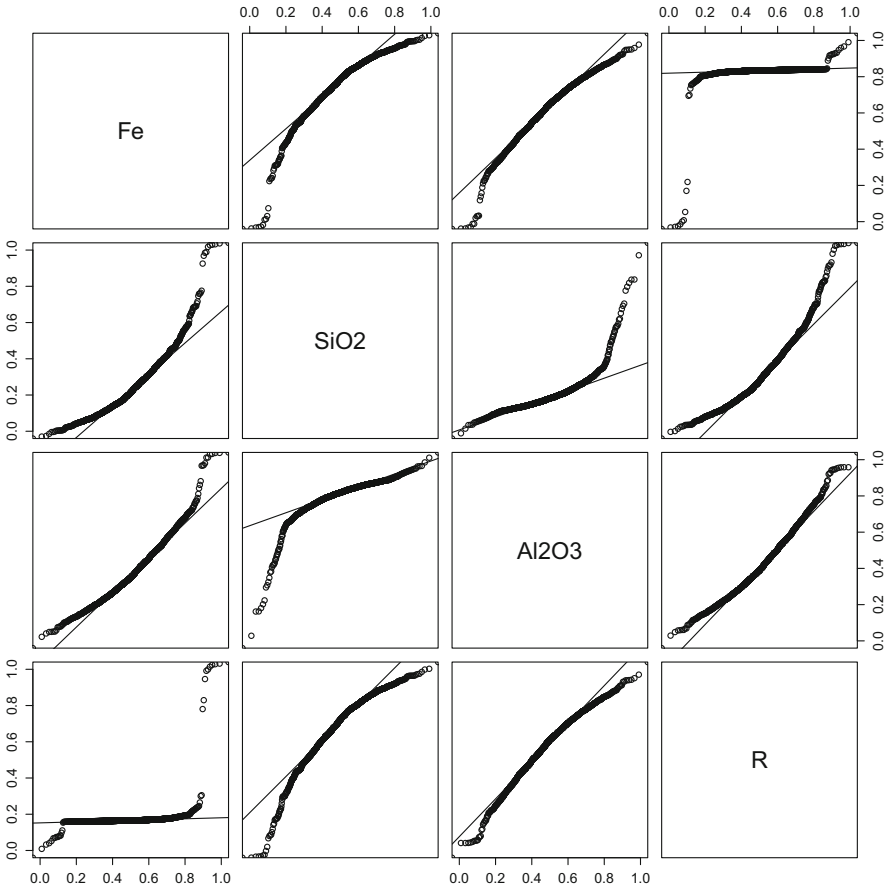

Fig. 3.9 Pairwise log-ratio pairs QQ-plots for the fit of a normal distribution to the subcomposition (Fe, SiO_2, Al_2O_3, R) of the Windarling data set. Thin line is the normal reference

for these tasks, albeit ignoring for the time being any issue of spatial dependence between the data.

The key idea of outlier counting, detection and classification relies on the existence of robust estimators for the central tendency and the spread form introduced earlier in this chapter. For a certain statistic θ, a *robust estimator* $\hat{\theta}_\alpha$ *with a breakpoint* α ($\alpha \in (0, 0.5)$) is one that does not change its value if we arbitrarily distort any percentage below $100\alpha\%$ of the observations (Rousseeuw & van Driessen, 1999). All statistics given in Sect. 3.2 have a breakpoint of $\alpha = 0$, that is they are non-robust. However, the package "`compositions`" provides robust versions of them, to be computed with the same commands given before, but with

the extra argument `robust="mcd"`

```
> mean(acomp(wind.compo), robust="mcd")

   Fe      P  SiO2 Al2O3      S     Mn     CL   LOI      R
0.651 0.001 0.016 0.008   BDL 0.001   BDL 0.038 0.284
attr(,"class")
[1] acomp

> variation(acomp(wind.compo), robust="mcd")

          Fe      P  SiO2 Al2O3      S     Mn     CL   LOI      R
Fe     0.000 0.161 0.570 0.720 0.372 0.701 0.226 0.157 0.000
P      0.161 0.000 0.740 0.806 0.541 0.728 0.390 0.212 0.159
SiO2   0.570 0.740 0.000 0.137 0.522 1.267 0.519 0.569 0.567
Al2O3  0.720 0.806 0.137 0.000 0.559 1.389 0.663 0.726 0.716
S      0.372 0.541 0.522 0.559 0.000 1.097 0.353 0.304 0.371
Mn     0.701 0.728 1.267 1.389 1.097 0.000 0.933 0.596 0.697
CL     0.226 0.390 0.519 0.663 0.353 0.933 0.000 0.306 0.225
LOI    0.157 0.212 0.569 0.726 0.304 0.596 0.306 0.000 0.156
R      0.000 0.159 0.567 0.716 0.371 0.697 0.225 0.156 0.000
```

This instructs the program to use the *minimum covariance determinant* estimator, an efficient algorithm that determines the smallest ellipsoid of the (ilr-transformed) data containing at least $100(1-\alpha)\%$ of the data. The centre of the ellipsoid is taken as the central tendency estimator, and the quadratic form describing the ellipsoid can be univocally converted into an estimate of the covariance matrix.

Once robust estimates of central tendency and spread are available, a useful task in outlier classification is the detection of possible minor subpopulations. The package provides an heuristic for that, `ClusterFinder1`; other may exist in other packages or in future versions of "`compositions`". `ClusterFinder1` spherifies the data via Eq. (3.7) with the robust mean and covariance matrix and then seeks to decompose the data in a mixture of multivariate normal models or model based cluster analysis

```
> wind.subcompo = wind.compo %>%
+   dplyr::select(Fe, SiO2, Al2O3, Mn) %>% acomp
> wind.cf1 = wind.subcompo %>% ClusterFinder1
> sort(wind.cf1$typeTbl, decreasing = T)

types
     1      3      2      8      4      5 single
  1532     34     12      8      7      5      2
```

A summary report of the results (provided in the element `typeTbl`) shows that 1532 observations appear to belong to the main population of the chosen subcomposition, with two smaller but perhaps significant groups (3 and 2, `single` displays how many observations appear to belong to a single-observation group), the rest belonging to potentially quite small groups. Alternatively, the robustly spherified scores can be compared with a reference distribution: if the p value or exceedance probability of this distribution above the norm of the spherified scores for a certain observation is "large enough" (say larger than 5%), then that particular observation

is considered non-outlying; otherwise, it is considered an outlier or extreme value. The difference between "(proven) outlier" and "extreme value" is whether that p value was corrected for multiple testing (proven outliers) or not (extreme values). A reference distribution typically used is the chi-square distribution with degrees of freedom equal to the number of coefficients of the spherified vectors (mostly $D - 1$ in the context of this book). The package "`compositions`" provides a second reference distribution, the empirical Mahalanobis norm distribution, which accounts for the fact that mean and covariance were estimated and are actually not known. This is packed in an omnibus function `OutlierClassifier1` with extra argument `type="grade"`

```
> wind.oc1g = OutlierClassifier1(wind.subcompo, type="grade")
> table(wind.oc1g, windarling$Lithotype)

wind.oc1g basalt goethite hematite schist
  ok           0      276      164      0
  extreme      1      539      550      7
  outlier      0       37       16     10
```

The result shows that 63 observations are clearly outliers, and 1097 more are extreme values or potential outliers. Interestingly, all schist samples (according to their Lithotype value) are outliers or extreme values.

Within a compositional context it is particularly interesting to consider the idea of single element "pollution": observations that qualify as outliers in the whole composition, but when one particular element of that composition is removed, they become regular samples in the resulting subcomposition. These calculations are done in the framework of empirical Mahalanobis norm distributions mentioned before with command `OutlierClassifier1` and extra argument `type="best"`

```
> wind.oc1b = OutlierClassifier1(wind.subcompo, type="best")
> table(wind.oc1b, windarling$Lithotype)

wind.oc1b basalt goethite hematite schist
    ok         1      820      720      7
    ?          0        6        3     10
    Fe         0        0        1      0
    SiO2       0       24        4      0
    Al2O3      0        2        0      0
    Mn         0        0        2      0
```

In this example, the results of these calculations indicate that out of the samples of hematite 1 has an anomalous Fe value only, and within hematite and goethite 28 become regular if SiO_2 is ignored. For Al_2O_3 there are 2 samples that become regular if this element is ignored and both are located within goethite. Finally, the results also suggest that 10 of the schist samples that were shown to be proven outliers are radically different, not just in one single component but in several. To obtain a report on all possible components that can be removed to make the subcomposition regular, the `type` is set to "all".

```
> wind.oc1a = OutlierClassifier1(wind.subcompo, type="all")
> table(wind.oc1a, windarling$Lithotype)
```

```
wind.oc1a basalt goethite hematite schist
     ok        1      820      720      7
     0000      0        6        3     10
     0001      0        0        2      0
     0100      0       20        3      0
     0110      0        6        1      0
     1000      0        0        1      0
```

In this case each datum gets a binary code of length equal to the number of components, here 4, with 1 in the ith position indicating that removal of component i will make the datum regular. The results of these calculations indicate that out of the samples of hematite 1 has an anomalous Fe (code: 1000) value only, and within hematite and goethite 23 become regular if SiO_2 (code: 0100) is ignored. There are 7 samples (6 in goethite and 1 in hematite) that become regular if SiO_2 and Al_2O_3 (code: 0110) are ignored and there are 2 samples that become regular if Mn is ignored.

A word of warning is relevant at this point: all these methods are intended for exploratory purposes, to understand and monitor the variability in the data set, and eventually to keep track of data quality issues. It is not legitimate to simply erase samples of the data set just because they are classified as outliers, not in classical statistical uses, but much less in geostatistical applications. The reasons will be discussed in Sect. 4.2.

Problems

3.1 Spherification and Invariance

Given a decomposition $\mathbf{S} = \mathbf{R} \cdot \mathbf{R}^t$, show that the spherified coefficients of Eq. (3.7) satisfy

$$d^2_{AM}(\mathbf{z}, \mathbf{z}'|\mathbf{S}) = d^2_{Eucl}(\boldsymbol{\eta}(\mathbf{z}), \boldsymbol{\eta}(\mathbf{z}')),$$
$$\langle \mathbf{z}, \mathbf{z}'|\mathbf{S}\rangle_{AM} = \left(\boldsymbol{\eta}(\mathbf{z}), \boldsymbol{\eta}(\mathbf{z}')\right).$$

3.2 Numerical and Graphical EDA for a Tellus Subcomposition

(a) Consider the subcomposition of the Tellus data set comprised of CaO, Al_2O_3, MgO, Fe_2O_3 and R. Determine graphical representations for the pairwise log-ratios of this subcomposition.
(b) Calculate the clr-PCA for the above subcomposition.
(c) Does the above subcomposition follow an additive logistic normal distribution?
(d) Does the subcomposition contain outliers?

3.3 Numerical and Graphical EDA for a NGSA Subcomposition

(a) Consider the subcomposition of the NGSA data set comprised of As, Bi and S. Produce graphical representations and numerical summaries for this subcomposition. Remove all observations where any of these variables is missing or below detection limit.
(b) Calculate the clr-PCA for the above subcomposition.
(c) Does the above subcomposition follow an additive logistic normal distribution?
(d) Does the subcomposition contain outliers?

References

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

Aitchison, J., & Greenacre, M. (2002). Biplots of compositional data. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 51*(4), 375–392.

Boogaart, K. G. v. d., & Tolosana-Delgado, R. (2013). *Analysing compositional data with R* (280 pp.). Heidelberg: Springer.

Filzmoser, P., & Hron, K. (2008). Outlier detection for compositional data using robust methods. *Mathematical Geosciences, 40*(3), 233–248.

Filzmoser, P., Hron, K., & Templ, M. (2009). Discriminant analysis for compositional data and robust estimation (27 pp.). Technical Report SM-2009-3, Department of Statistics and Probability Theory, Vienna University of Technology, Austria.

Grunsky, E. C., & Caritat, P. d. (2020). State-of-the-art analysis of geochemical data for mineral exploration. *Geochemistry: Exploration, Environment, Analysis, 20*(2), 217–232.

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. (2008). A critical approach to probability laws in geochemistry. *Mathematical Geosciences, 40*(5), 489–502.

McKinley, J. M., Grunsky, E. C., & Mueller, U. (2018). Environmental monitoring and peat assessment using multivariate analysis of regional-scale geochemical data. *Mathematical Geosciences, 50*(2), 235–246.

Pawlowsky-Glahn, V., Egozcue, J. J., & Tolosana-Delgado, R. (2015). *Modeling and analysis of compositional data* (272 pp.). Chichester, UK: John Wiley & Sons.

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

Rousseeuw, P., & van Driessen, K. (1999). A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics, 41*, 212–223.

Székely, G. J., & Rizzo, M. L. (2005). A New Test for Multivariate Normality. *Journal of Multivariate Analysis, 93*(1), 58–80.

Chapter 4
Exploratory Spatial Analysis

Abstract In this chapter the tools for spatial exploratory analysis are provided. These include data postings, swathplots and experimental variograms.

4.1 The R-packages "sp", "gstat" and "gmGeostats"

The **R**-package "sp" provides a set of data containers designed for spatial objects and for spatially dependent data. These include containers for much of the same topologies that can be captured with 2D-geographic information systems (GIS): locations, lines and polygons for vector-based GIS, and grids and pixels for raster-based GIS. The package is loaded via

```
> library("sp")
```

For the goals of this book, we will only briefly discuss data containers for points and grids; readers are referred to the excellent book by Bivand et al. (2013) for an in-depth discussion of all spatial object classes. Class "SpatialPoints" is designed to gather sets of locations dispersed in space, while "SpatialPointsDataFrame" extends it to contain data associated with these locations. Similarly, "SpatialGrid" captures a fully regular, extended by "SpatialGridDataFrame" to contain the data associated with the grid locations. An intermediate class is "SpatialPixels" (with associated data container "SpatialPixelsDataFrame") devised to contain just a subset of locations of the regular grid. In all three cases, the number of points of a set must correspond to the number of rows of the associated data set. A "SpatialGrid" can be comfortably created with the specification of a grid topology by means of the function "GridTopology", which requires the coordinates of the upper left top cell, the cell size and the number of cells on each direction. An example of the definition of a grid topology is given on page 223. The data set contained in an object of class "Spatial***DataFrame" can be extracted with the syntax `slot(object, "data")`. The spatial coordinates of the locations contained in any of the objects mentioned can be obtained with the syntax `coordinates(object)`.

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_4

Package "gstat" provides a comprehensive set of tools for the treatment of geostatistical data in 1, 2 or 3 dimensions. It comprises variogram calculation and modelling and the computation of estimates or simulated realisations of the data on a user defined grid.

The package is loaded via

```
> library("gstat")
```

and spatial data are specified as a "gstat" object by defining the name, the object, the variables to be modelled, the coordinates, the calculation parameters for experimental variograms and associated models along with the relationship between the data and their coordinates and the grid. Further estimation and simulation details need to be specified. A scheme of the workflow and functions provided by "gstat" is provided in Sect. 5.3.1 and the Appendix B. Most of the functions of "gstat" require having available spatial coordinates and data from the observations. These can be provided bundled in "SpatialPointsDataFrame" objects, or else separately. This book mostly follows the second approach, specifying both coordinates and data.

The companion package to this book, "gmGeostats" acts as a bridge between packages "sp", "gstat" and "compositions". The functionalities of "gmGeostats" are similar to those of "gstat", but are devised for a fully multivariate modelling experience. Data in "gmGeostats" are organised in layers of information, which can be multivariate or univariate in nature, and each will have its own scale (i.e. its way of comparing values, as explained in Chap. 2). These multi-layered data sets can be seamlessly linked with coordinates in "Spatial***DataFrame" objects. As in "gstat", a geostatistical model is constructed by binding these spatial data frames with certain model parameters and certain method parameters. Several geostatistical techniques can then be obtained with a systematic combination of these model and method parameters. Details on the way this is done will be given throughout the book, in the relevant chapters.

4.2 Spatial Data Analysis

Following on from an exploratory compositional descriptive statistics, the spatial characteristics of the data need to be established. We therefore now need to work in the framework of regionalised compositions as now the spatial coordinates of the data will need to be considered along with compositional properties. In this section, tools for exploratory spatial analysis are discussed. These include different types of spatial maps and plots of the variables of interest against the coordinate directions. These will be illustrated using the subcomposition {Fe, Al_2O_3, SiO_2, Mn, P)}. This subcomposition needs to be reclosed prior to the start of any analysis, as shown in the following chunk:

```
> data("Windarling",package="gmGeostats")
> windarling=Windarling
```

```
> wind.compo = windarling %>% dplyr::select(Fe, Al2O3, SiO2, Mn, P)
> wind.compo$R = 1-rowSums(wind.compo)
```

4.2.1 Maps and Plots

Different types of spatial maps can be used to visualise the spatial distribution of a variable of interest. These include *data postings*, where a map of the coordinates of the data is generated with location markers either colour-coded or sized proportionally to the magnitude of the given data. They allow the joint visualisation of the composition from a spatial perspective. A set of simple colour-coded maps can be obtained with the command `pairsmap` from package `gmGeostats` as follows :

```
> wind.coords = as.matrix(windarling[,c("Easting","Northing")])

> pairsmap(data=wind.compo, loc=wind.coords, cexrange=c(1,1)*0.75,
+          mfrow=c(6,1), foregroundcolor=NA)
```

If a spatial map with markers sized proportionally to the magnitude of the values (Fig. 4.1) is desired, the variable "cexrange" needs to be adjusted to allow for different symbol sizes.

```
> pairsmap(data=wind.compo, loc=wind.coords, cexrange=c(1,3)*0.5,
+          mfrow=c(6,1), foregroundcolor=NA)
```

In either case it is clear that the spatial patterns differ from variable to variable with Fe generally having higher values in the north of the bench, while the SiO_2 data have larger values in the south and a region of particularly high values in the central part of the south east. There is also a low level Mn region in the centre of the eastern bench of Windarling. One could think of this part as potentially high quality Fe ore, with beneficial Fe/Mn ratios. However, the SiO_2 values of that region are very high, suggesting a dilution effect of bulk chemistry by SiO_2, irrespective of the ratio to Fe, again highlighting the compositional nature of the data. To obtain a spatial map of the alr-transformed data, the input variables (Fig. 4.2) need to be recomputed in alr-coordinates.

```
> pairsmap(data=alr(wind.compo), loc=wind.coords,
+          cexrange=c(1,3)*0.5, mfrow=c(5,1), foregroundcolor=NA)
```

In the case of the Windarling data, the differences between the spatial maps of the raw composition and its coordinates in alr-space relative to the variable R are not very pronounced which is a consequence of a small IQR, and of the fact that the bench represents part of a high grade iron ore deposit.

The spatial behaviour of the data can be studied further via *swath plots* of the pairwise log-ratios of the composition. Conventional swath plots can be computed by splitting the coordinates in the X or Y direction into classes, and producing means (even boxplots) of the variable under consideration for those classes. The result

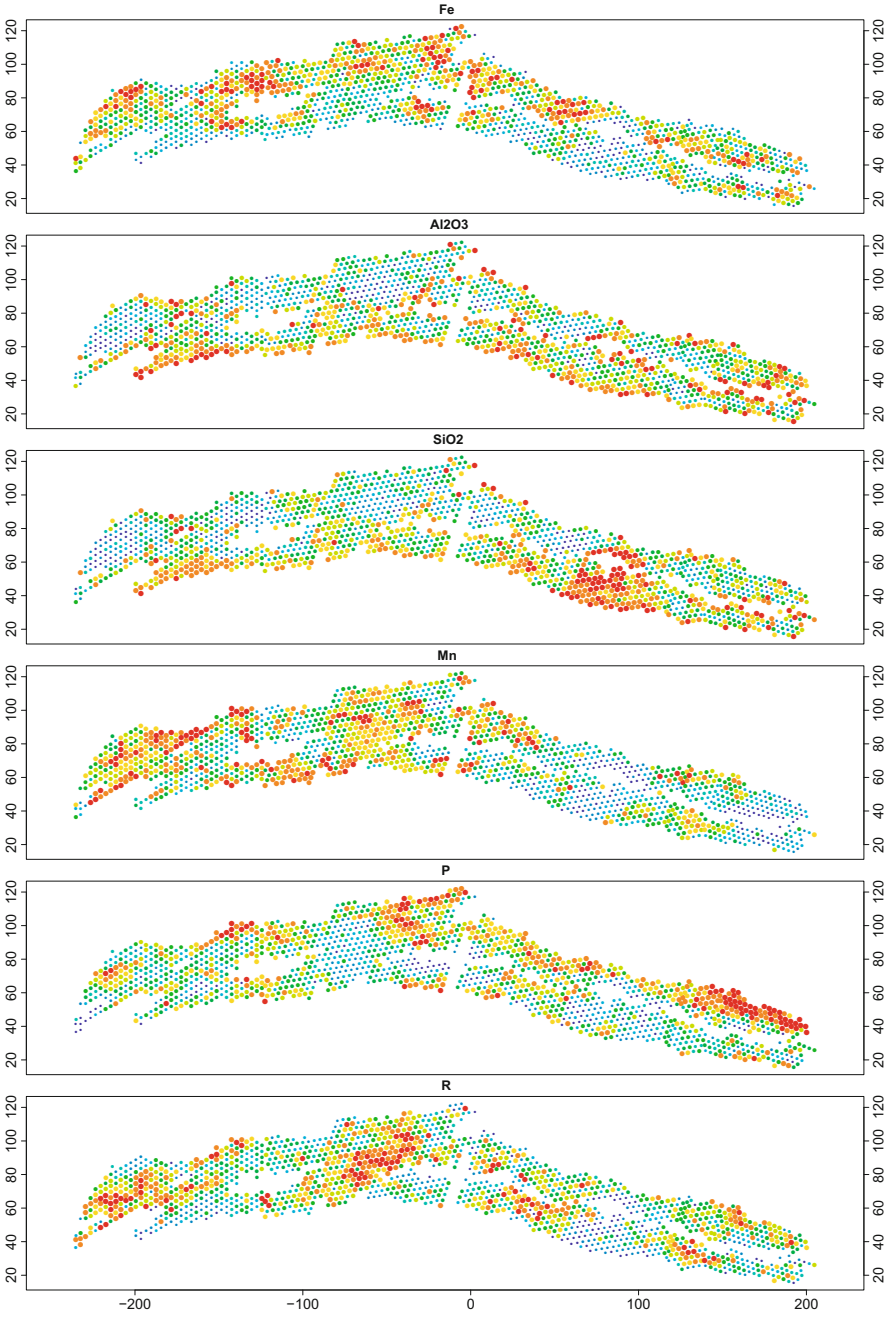

Fig. 4.1 Spatial maps of the Windarling composition with proportional symbols

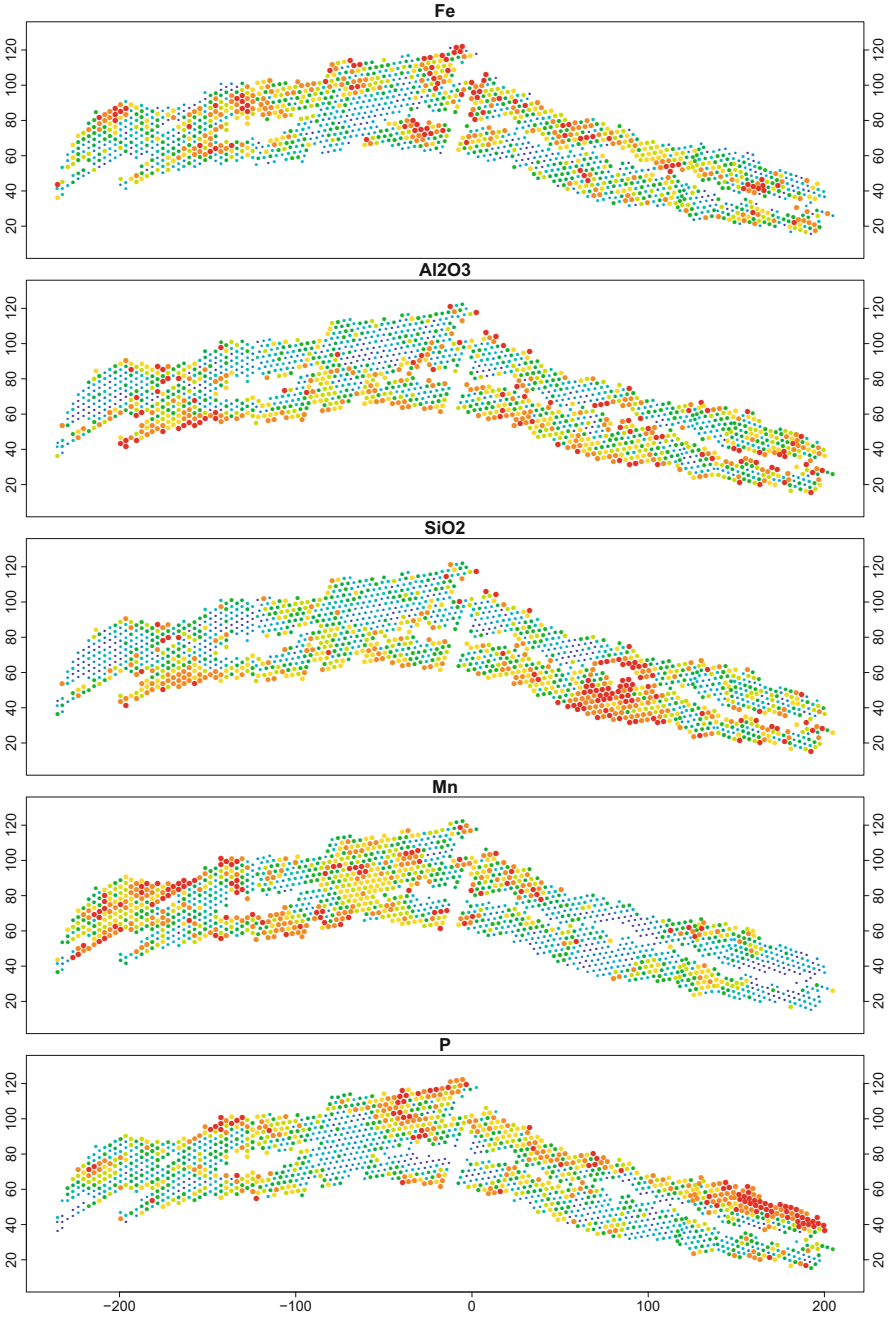

Fig. 4.2 Spatial maps of the Windarling composition in alr-coordinates relative to the variable R

is usually a rough curve of the behaviour of the values relative to the coordinate direction and experimentation is required to obtain a plot that is interpretable. To avoid this complication this functionality is programmed with a local spline regression in this package and moving windows are used to avoid discontinuities.

```
> swath(acomp(wind.compo), wind.coords[,"Easting"], col=8,
+           xlab="Easting")

> swath(acomp(wind.compo), wind.coords[,"Northing"], col=8,
+           xlab="Northing")
```

The plots can be generated either with a common vertical scale by rows (the default for a composition), or with each plot scaled individually (with argument `commonScale = FALSE`). For the Windarling data, these diagrams show no significant trend in the EW direction (Fig. 4.3), but some evidence of a difference in the NS direction (Fig. 4.4), especially for ratios such as Fe/SiO_2, Mn/SiO_2 or Mn/Al_2O_3.

A final aspect that should be considered while exploring the data maps is the spatial arrangement of outliers and subpopulations. The goal is here to evaluate the possibility that the spatial domain is not well defined, and that samples at the boundaries may need to be removed. This analysis is best done with dot maps

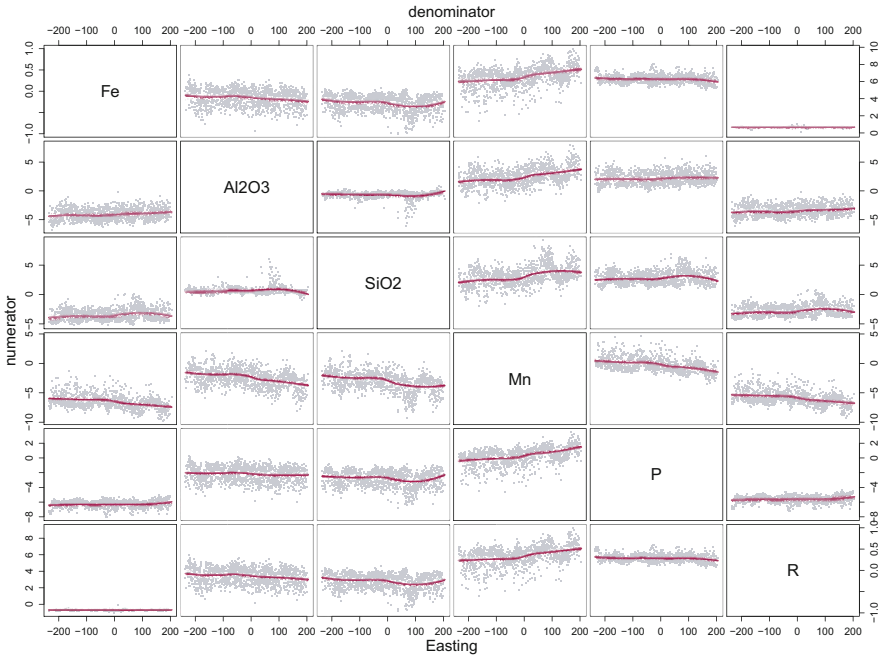

Fig. 4.3 Swath plots of pairwise log-ratios of Windarling data in the EW direction

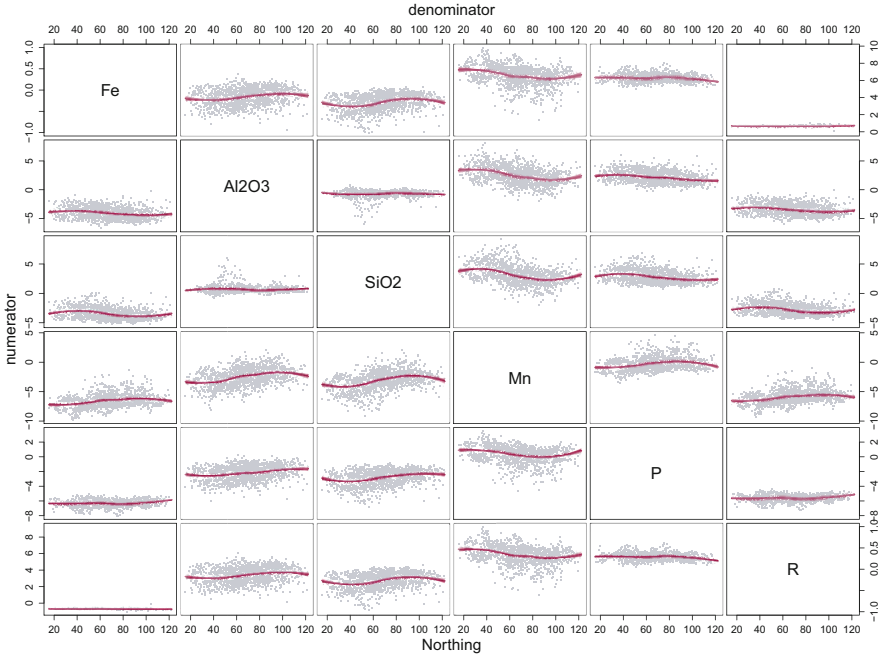

Fig. 4.4 Swath plots of pairwise log-ratios of Windarling data in the NS direction

coloured by a categorical variable of interest, for instance,

```
> par(mfrow=c(2,1))
> plot(wind.coords, col=windarling$Lithotype, asp=1)
> legend("topright", fill=1:4,
+         legend = levels(windarling$Lithotype))
> plot(wind.coords, col=c("black","green","red")
+       [as.integer(wind.oc1g)], asp=1)
> legend("topleft", fill=c("black","green","red"),
+         legend=levels(wind.oc1g))
```

The resulting figure (Fig. 4.5) shows that basalt and schist samples are all located at the boundaries, mostly at the external boundaries. Even the three located in the interior part of the domain do actually belong to unsampled regions inside the bench, which probably are not to be exploited. Consequently, it is legitimate to remove these samples from the data set. Any categorical variable available can be analysed in a similar manner, for instance, with the result of an outlier exploratory analysis from Sect. 3.5 (Fig. 4.5 bottom). In this case, we see that there are two kinds of true outliers: There is a spatial cluster of them in the middle of the eastern part of the bench, and additionally a dozen isolated outliers around the boundary. These two kinds should be considered separately. Depending on the goals of the analysis, the inner cluster could be treated as a sub-domain, and analysed separately. In contrast, the isolated outliers at the boundaries could just be removed from further analysis.

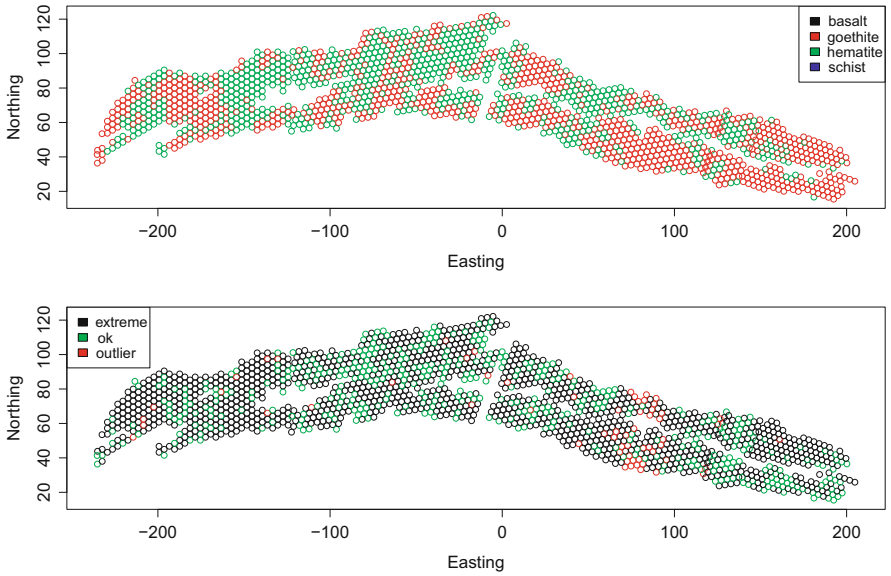

Fig. 4.5 Maps of sample locations, with colours according to lithotype (upper panel) and degree of anomality in the subcomposition (Fe, SiO_2, Al_2O_3, Mn) (lower panel)

In this particular case, we will not exclude these on the basis of the outlier analysis alone, but because they mostly overlap with the schist samples. We therefore exclude them according to their geology

```
> goodLitho = windarling$Lithotype %in% c("hematite","goethite")
> wind.compo = wind.compo[goodLitho,]
> wind.coords = wind.coords[goodLitho,]
> windarling = windarling[goodLitho,]
```

4.3 Variograms

Two different tools have been proposed for characterising the spatial dependence structure of a regionalised compositional data set (or its theoretical counterpart, a compositional random field). We can either use classical direct and cross variograms of the log-ratio transformed data (in analogy to $\boldsymbol{S}$); or the set of pairwise log-ratio variograms (analogous to the variation matrix $\mathbf{T}$). Both were defined for the first time by Pawlowsky (1986). The properties, advantages and inconveniences of each tool are discussed in detail in Pawlowsky-Glahn and Olea (2004) and Tolosana-Delgado et al. (2011). This section focusses on the most relevant ones from a practical point of view, and we first of all recall the definition of variograms for univariate and multivariate data in general.

4.3.1 Raw Variograms

Let $\{Y(\mathbf{x}) : \mathbf{x} \in \mathscr{A}\}$ be a random function defined on a subset $\mathscr{A}$ of 2 or 3 dimensional real space. The random function is said to be *second order stationary* (Chilès & Delfiner, 2012; Goovaerts, 1997), if the following conditions are satisfied:

- The expected value is independent of location: $\mathrm{E}[Y(\mathbf{x})] = m$.
- The covariance function $C(\mathbf{h}) := \mathrm{cov}[Y(\mathbf{x}+\mathbf{h}), Y(\mathbf{x})]$ exists and depends only on the lag vector $\mathbf{h}$.

The assumption of second order stationarity is commonly weakened to *intrinsic stationarity* (Goovaerts, 1997) which is nothing other but second order stationarity for the increments $(Y(\mathbf{x}+\mathbf{h}) - Y(\mathbf{x}))$ where, as before, $\mathbf{x} \in \mathscr{A}$ and $\mathbf{h}$ is a lag vector.

In the multivariate case, the definitions of second order stationarity and intrinsic stationarity are analogous except that the expected value of the random function is a vector $\mathbf{m} = \mathrm{E}[\mathbf{Y}(\mathbf{x})]$ and the *covariance function* $\mathbf{C}(\mathbf{h})$ is matrix-valued. In either case the choice of different vectors for $\mathbf{h}$ gives rise to either a function of $\mathbf{h}$ or a matrix-valued function of $\mathbf{h}$, which is used to construct a model of the spatial continuity of the random function. In the intrinsic case, the matrix-valued function

$$\boldsymbol{\Gamma}(\mathbf{h}) := \boldsymbol{\Gamma}(\mathbf{x}, \mathbf{x}+\mathbf{h}) = \frac{1}{2}\mathrm{var}[(\mathbf{Y}(\mathbf{x}+\mathbf{h}) - \mathbf{Y}(\mathbf{x}))] \tag{4.1}$$

is finite and depends only on the magnitude and orientation of the displacement vector $\mathbf{h}$, but not on the location $\mathbf{x}$. If the intrinsic hypothesis is satisfied, then the function $\boldsymbol{\Gamma}(\mathbf{h})$ exists and is called a *variogram function*.

The experimental counterpart of the variogram function is defined based on a regionalised sample $\{\mathbf{y}_i := \mathbf{y}(\mathbf{x}_i) : \mathbf{x}_i \in \mathscr{A}, i = 1, \ldots, N\}$ of observed values in the study region. Experimental versions for the covariance function (either univariate or multivariate) are seldom available.

Assuming that at each location, every component of the vector-valued regionalised vector was observed, (i.e. no missing values, not-measured, values below detection limit, etc.), then for a given lag distance class $\mathbf{h}_k$, the experimental variogram is defined as

$$\hat{\boldsymbol{\Gamma}}(\mathbf{h}_k) = \frac{1}{2N(\mathbf{h}_k)} \sum_{\mathbf{x}_n - \mathbf{x}_m \simeq \mathbf{h}_k} (\mathbf{y}_n - \mathbf{y}_m) \cdot (\mathbf{y}_n - \mathbf{y}_m)^t, \tag{4.2}$$

where $N(\mathbf{h}_k)$ denotes the number of pairs separated by the lag vector $\mathbf{h}_k$. Lag vectors are usually taken as multiples of a lag vector that is based on nearest neighbour separations, i.e. $\mathbf{h}_k = k\mathbf{h}_1$ where $\mathbf{h}_1$ is a vector whose length reflects a specified nearest neighbour distance.

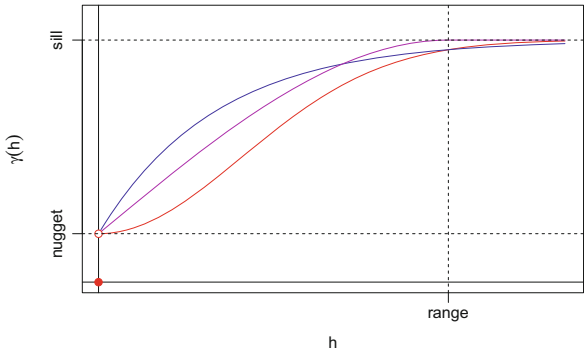

Fig. 4.6 Common elements for the qualitative description of a variogram

4.3.2 Practical Aspects

Geostatisticians typically describe their empirical variograms with the help of four conceptual *parameters* (Fig. 4.6): sill, range, nugget and anisotropy. Most variograms approximate a plateau value at long lags: That value is called the *variogram sill*. The lag at which this sill is achieved is called the *variogram range*. Additionally, the variogram may or may not approach 0 for small lags: If this is the case, the limiting value of the variogram at lag 0 is called the *nugget*. More generically, it is important to consider the behaviour of the variogram around the origin and at long distances. The behaviour at the origin can always be approximated by a curve h^α of the lag distance $h = \|\mathbf{h}\|$, with $\alpha \in (0, 2)$. For $\alpha \to 0$ the variogram becomes discontinuous at the origin, giving rise to the nugget effect. For $\alpha \to 2$ the variogram becomes parabolic at the origin, indicating a very smooth random function. For $\alpha \approx 1$ the variogram is linear at the origin, which implies that the random function is continuous, but not necessarily differentiable. At long distances, the variogram can stabilise around the sill, it can oscillate around it with a fixed period, or else grow towards ∞ in a way proportional to h^α: In this last case, a parabolic behaviour indicates the presence of a spatial *trend*, an issue that will be discussed in the following chapters.

To explore the spatial continuity of the Windarling data first a suitable lag spacing for computing variogram values needs to be determined. To do so, a histogram of the sample separations is generated. Since the spatial extent in the EW direction is 440 m and in the NS direction 60 m, 125 classes will be used for the histogram of distances.

```
> dd=dist(wind.coords)

> ddm = as.matrix(dd)
> summary(c(ddm))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0      61     125     142     208     440

> diag(ddm) = NA
> summary(apply(ddm, 1, min, na.rm=T))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.79    3.47    3.47    3.41    3.47    5.55
```

```
> range(wind.coords[,"Easting"])
```

```
[1] -235  205
```

```
> range(wind.coords[,"Northing"])
```

```
[1]  15.5 122.2
```

From the calculations of the nearest neighbours for each row, it is clear that a lag separation in the vicinity of 3.5m will be suitable for computing experimental variograms and the total separation distance in the EW direction should be app. 200 m, while in the NS direction we need 50 m, a check of the sample map suggests to even reduce that to 30m (half of the typical width).

The package "gstat" offers a simple way of computing raw variograms. As a first step, a data container for the spatial data set needs to be created. The function to do so is also called gstat. The data container "gstat" must be given a name for the variable being analysed, the data itself, and two formulas, formula and locations identifying the variable to be analysed and the variables to take as spatial coordinates. All names of variables occurring in each of these formulas must exist in the data set provided. The meaning of ∼1 in the formula argument will be discussed in the following chapters. The experimental variogram is calculated via the command and requires specification of the "gstat" object.

```
> wind.Fe.gg = gstat(id="Fe", formula=Fe~1, data = windarling,
+                    locations=~Easting+Northing)
> wind.Fe.vg = variogram(wind.Fe.gg)
> plot(wind.Fe.vg)
```

Other arguments are accessory, such as the total separation distance up to which the experimental direct and cross variograms are to be computed (cutoff) and the lag width (width). The experimental variogram may then be plotted using the plot command. For directional variograms in 2D problems, the reference directions in the plane need to be specified with argument alpha. In 3D the directions are specified with two parameters, alpha and beta. Anisotropy will be discussed in detail in Sect. 4.3.6.

4.3.3 Variograms for Compositional Data: Coordinate Variography

Population variograms for compositional data can be derived from their coordinate representation, and are referred to as *coordinate variograms*. Let $\mathbf{Z}(\mathbf{x})$ be a compositional random function. Let $\{\mathbf{z}_i =: \mathbf{z}(\mathbf{x}_i) : \mathbf{x}_i \in \mathscr{A}, i = 1, \ldots, N\}$ be a compositional regionalised sample observed in the study region. Under the

assumption of intrinsic stationarity of order 2, its variation in space can be described by

$$\mathbf{m}(\mathbf{h}) = \mathrm{E}[\mathrm{glr}(\mathbf{Z}(\mathbf{x}+\mathbf{h})) - \mathrm{glr}(\mathbf{Z}(\mathbf{x}))] = \mathbf{0} \tag{4.3}$$

and the matrix-valued function

$$\mathrm{var}[\mathrm{glr}(\mathbf{Z}(\mathbf{x}+\mathbf{h})) - \mathrm{glr}(\mathbf{Z}(\mathbf{x}))] =: 2\boldsymbol{\Gamma}(\mathbf{x}, \mathbf{x}+\mathbf{h}) = 2\boldsymbol{\Gamma}(\mathbf{h}). \tag{4.4}$$

Similarly for a given lag distance class $\mathbf{h}_k$, the experimental coordinate variogram is defined as

$$\hat{\boldsymbol{\Gamma}}(\mathbf{h}_k) = \frac{1}{2N(\mathbf{h}_k)} \sum_{\mathbf{x}_n - \mathbf{x}_m \simeq \mathbf{h}_k} (\mathrm{glr}(\mathbf{z}_n) - \mathrm{glr}(\mathbf{z}_m)) \cdot (\mathrm{glr}(\mathbf{z}_n) - \mathrm{glr}(\mathbf{z}_m))^t. \tag{4.5}$$

Here glr denotes one of the log-ratio transformations in Chap. 2. The magnitudes of the lag vectors depend on the data and need to be determined from the sample separations and the spatial extent of the study region.

Both packages "gmGeostats" and "gstat" provide tools for working with multivariate data, and in particular with compositions. Package "gstat" requires the transformation of the composition to log-ratio scores, and then creation of a "gstat" object by manually adding layers of information to it

```
> wind.alr = alr(wind.compo)
> wind.alr.gg = gstat(id="alr1", formula=alr1~1,
+           data = cbind(wind.coords, wind.alr),
+                 locations=~Easting+Northing)
> wind.alr.gg = gstat(wind.alr.gg, id="ilr2", formula=alr1~1,
+           data = cbind(wind.coords, wind.alr),
+                 locations=~Easting+Northing)
>
```

and so on. This is tedious, and it is easy to type something wrong. For instance, these commands above would fail, because the column names of `wind.alr` do not correspond to the names of the variables given in the formula. Instead, it is recommended to create the data object by means of package "gmGeostats", which provides dedicated functions for establishing multivariate spatial data objects. In the case of compositional data, the command is

```
> library(gmGeostats)
> gg = make.gmCompositionalGaussianSpatialModel(
+                              data, coords, V, prefix, model, ng)
```

where `data` denotes the regionalised composition, `coords` contains the spatial coordinates and `V` specifies the primary log-ratio representation to be used for the data (defaults to ilr). Argument `prefix` can be used to generate user-friendly variable names after the transformation. Arguments `model` and `ng` (and many more not shown here) can be used for controlling the underlying statistical model and the method of analysis, and will be described in the next chapters. The output

of this function is a compact object of class "gmSpatialModel", that forms the cornerstone of this package usage. Such objects can easily be converted to a multivariate "gstat" object with the command

```
> gs = as.gstat(gg)
```

Computation of variograms is then a matter of using the function variogram on either of these two objects, both work equivalently with the command variogram, as explained for univariate cases in Sect. 4.3.1,

```
> wind.alr.gg =
+   make.gmCompositionalGaussianSpatialModel(
+     wind.compo, wind.coords, V = "alr")
> wind.alr.vg = variogram(wind.alr.gg)
> summary(wind.alr.vg)
```

```
       np              dist           gamma            dir.hor
 Min.   : 18872   Min.   :  6.8   Min.   :-0.188   Min.   :0
 1st Qu.: 50307   1st Qu.: 35.2   1st Qu.:-0.017   1st Qu.:0
 Median : 88738   Median : 75.5   Median : 0.008   Median :0
 Mean   : 81630   Mean   : 75.6   Mean   : 0.197   Mean   :0
 3rd Qu.:100614   3rd Qu.:115.7   3rd Qu.: 0.542   3rd Qu.:0
 Max.   :140868   Max.   :145.8   Max.   : 1.002   Max.   :0

    dir.ver          id
 Min.   :0   alr1.alr5: 15
 1st Qu.:0   alr2.alr5: 15
 Median :0   alr3.alr5: 15
 Mean   :0   alr4.alr5: 15
 3rd Qu.:0   alr5     : 15
 Max.   :0   alr1.alr4: 15
             (Other)  :135
```

4.3.4 *Variograms for Compositional Data: Variation-Variograms*

In addition to the coordinate variograms for a regionalised composition, it is possible to compute the population and experimental variograms for all pairwise log-ratios. This construct leads to the *variation-variogram*, known in Pawlowsky-Glahn and Olea (2004) as lr-autocovariance. In terms of the population, it is given by $\mathbf{T}(\mathbf{x}, \mathbf{x}+\mathbf{h}) = \mathbf{T}(\mathbf{h}) = [t_{ij}(\mathbf{h})]$, with elements

$$t_{ij}(\mathbf{h}) =: \mathrm{var}\left[\ln\frac{Z_i(\mathbf{x}+\mathbf{h})}{Z_j(\mathbf{x}+\mathbf{h})} - \ln\frac{Z_i(\mathbf{x})}{Z_j(\mathbf{x})}\right], \tag{4.6}$$

while the experimental variation-variogram is the matrix-valued function $\hat{\mathbf{T}}(\mathbf{h}_k) = [\hat{t}_{ij}(\mathbf{h}_k)]$, with elements

$$\hat{t}_{ij}(\mathbf{h}_k) = \frac{1}{2N(\mathbf{h}_k)} \sum_{\mathbf{x}_n - \mathbf{x}_m \simeq \mathbf{h}_k} \left(\ln \frac{z_i(\mathbf{x}_n)}{z_j(\mathbf{x}_n)} - \ln \frac{z_i(\mathbf{x}_m)}{z_j(\mathbf{x}_m)} \right)^2 . \tag{4.7}$$

The packages "`compositions`" and "`gmGeostats`" provide a single function for computing variation-variograms for the data: `logratioVariogram`. This function must be given a regionalised composition, either in the form of a "`gmSpatialModel`" object, or alternatively by providing a composition and its spatial coordinates separately. Additionally, several arguments can be provided to control maximum spatial distance to consider, or the width in radius and directional tolerance to consider. For the Windarling data set, a total separation distance of 200 m is chosen and the number of lags is set to 60.

```
> wind.pwlr.vg = logratioVariogram(data = acomp(wind.compo),
+                          loc=wind.coords, nbins=60, maxdist=200)
```

The plot of the experimental variation-variogram is shown in Fig. 4.7. Each column or row of the variation-variogram plot represents the variograms in alr-coordinates relative to the corresponding element indicated on the main diagonal.

```
> plot(wind.pwlr.vg)
```

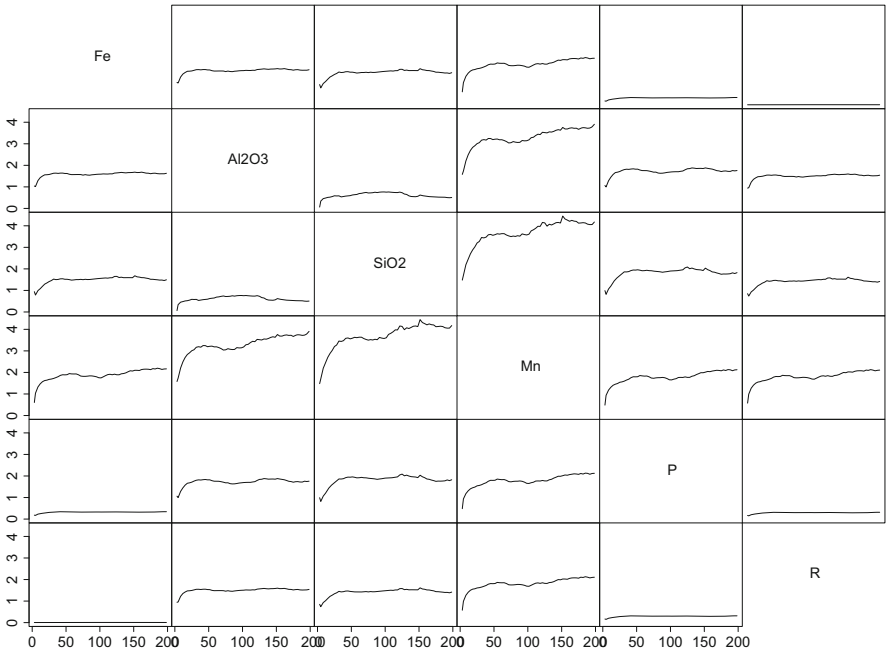

Fig. 4.7 Experimental variation-variograms of Windarling data

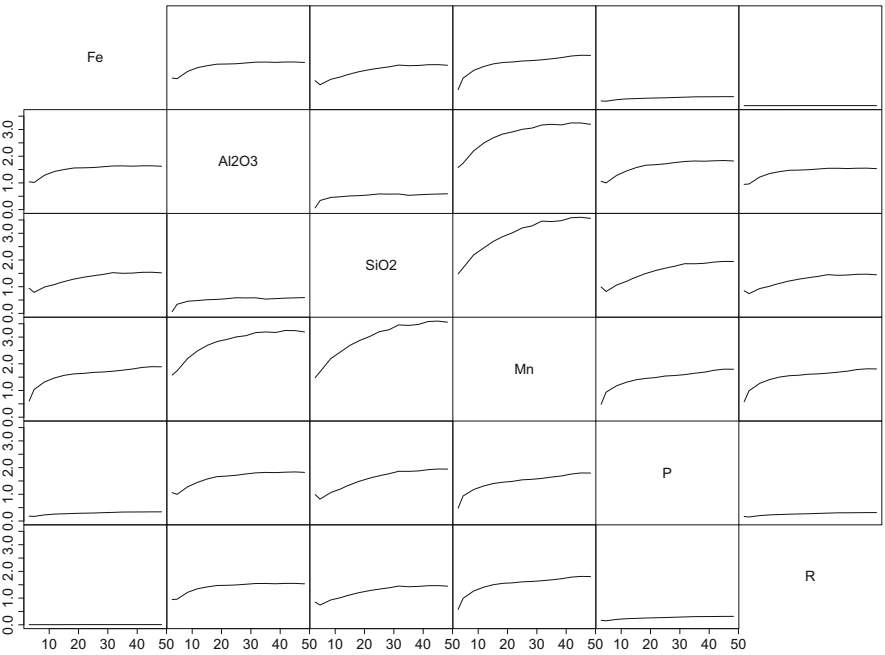

Fig. 4.8 Experimental variation-variograms of Windarling data computed for a total separation distance of 50m at a nominal spacing of 2.5m

From the experimental semivariograms of the pairwise log-ratios, it is clear that most pairs reach their sill at a range between 20m and 50m. It therefore makes sense to recalculate the variation-variogram for that overall distance (Fig. 4.8).

```
> wind.pwlr.vg = logratioVariogram(acomp(wind.compo),
+                          loc=wind.coords, maxdist=50, nbins=15)
> plot(wind.pwlr.vg)
```

4.3.5 Relationships Between Structural Functions

Pawlowsky-Glahn and Olea (2004) propose several structural functions, which are essentially classical variograms and covariance functions defined on either clr-transformed data, or alr-transformed data. In this section, these definitions are summarised and adapted to the case of an arbitrary log-ratio transformed regionalised composition.

Denote by $\mathbf{Y}(\mathbf{x}) = \text{glr}(\mathbf{Z}(\mathbf{x})) = \ln \mathbf{Z}(\mathbf{x}) \cdot \boldsymbol{\Psi}^t$ the scores obtained by transforming the regionalised composition $\mathbf{Z}(\mathbf{x})$ through a suitable full rank log-ratio transformation, with inverse $\mathbf{Z}(\mathbf{x}) = \mathscr{C}\left[\exp(\mathbf{Y}(\mathbf{x}) \cdot \boldsymbol{\Phi})\right]$ as described on page 18. For any choice of log-ratio transformation, a matrix-valued variogram or covariance

function can be defined, as shown in Sect. 4.3.3. Under these circumstances, the following equivalences hold (Tolosana-Delgado et al., 2019):

- For a given log-ratio transformation, its variogram and covariance function satisfy

$$\boldsymbol{\Gamma}(\mathbf{h}) = \mathbf{C}(\mathbf{0}) - \frac{1}{2}(\mathbf{C}(\mathbf{h}) + \mathbf{C}(-\mathbf{h}))\,;$$

  thus, for a symmetric covariance function, where $\mathbf{C}(\mathbf{h}) = \mathbf{C}(-\mathbf{h})$, the log-ratio variogram and the log-ratio covariance function are equivalent, and they also fulfil $\mathbf{C}(\mathbf{h}) = \mathbf{C}(\mathbf{0}) - \boldsymbol{\Gamma}(\mathbf{h})$.
- The covariance functions of variograms and covariance functions expressed in any arbitrary full rank log-ratio representation are linked to those of the clr-transformed composition by

$$\begin{aligned} \mathbf{C}^{\psi}(\mathbf{h}) &= \boldsymbol{\Psi} \cdot \mathbf{C}^{c}(\mathbf{h}) \cdot \boldsymbol{\Psi}^{t}, \quad \mathbf{C}^{c}(\mathbf{h}) = \boldsymbol{\Phi}^{t} \cdot \mathbf{C}^{\psi}(\mathbf{h}) \cdot \boldsymbol{\Phi}\,, \\ \boldsymbol{\Gamma}^{\psi}(\mathbf{h}) &= \boldsymbol{\Psi} \cdot \boldsymbol{\Gamma}^{c}(\mathbf{h}) \cdot \boldsymbol{\Psi}^{t}, \quad \boldsymbol{\Gamma}^{c}(\mathbf{h}) = \boldsymbol{\Phi}^{t} \cdot \boldsymbol{\Gamma}^{\psi}(\mathbf{h}) \cdot \boldsymbol{\Phi}\,. \end{aligned} \tag{4.8}$$

- These expressions simplify in the case that an ilr-transformation is used

$$\mathbf{C}^{i}(\mathbf{h}) = \boldsymbol{\Psi} \cdot \mathbf{C}^{c}(\mathbf{h}) \cdot \boldsymbol{\Psi}^{t}, \quad \mathbf{C}^{c}(\mathbf{h}) = \boldsymbol{\Psi}^{t} \cdot \mathbf{C}^{i}(\mathbf{h}) \cdot \boldsymbol{\Psi}\,,$$

  where the superscripts c and i relate to the clr and ilr transformations, respectively.
- Analogously, the corresponding variograms are related by

$$\boldsymbol{\Gamma}^{i}(\mathbf{h}) = \boldsymbol{\Psi} \cdot \boldsymbol{\Gamma}^{c}(\mathbf{h}) \cdot \boldsymbol{\Psi}^{t}, \quad \boldsymbol{\Gamma}^{c}(\mathbf{h}) = \boldsymbol{\Psi}^{t} \cdot \boldsymbol{\Gamma}^{i}(\mathbf{h}) \cdot \boldsymbol{\Psi}\,. \tag{4.9}$$

- Log-ratio variograms and the variation-variogram are related by

$$\boldsymbol{\Gamma}(\mathbf{h}) = -\frac{1}{2}\boldsymbol{\Psi} \cdot \mathbf{T}(\mathbf{h}) \cdot \boldsymbol{\Psi}^{t} \tag{4.10}$$

  with an inverse relationship between variation-variograms and clr-variograms

$$\mathbf{T}(\mathbf{h}) = \mathbf{1}_{(D,D)}\mathrm{diag}(\boldsymbol{\Gamma}^{c}(\mathbf{h})) + \mathrm{diag}(\boldsymbol{\Gamma}^{c}(\mathbf{h}))\mathbf{1}_{(D,D)} - 2\boldsymbol{\Gamma}^{c}(\mathbf{h}), \tag{4.11}$$

  where $\mathbf{1}_{(D,D)} = \mathbf{1}_D\mathbf{1}_D^t$ is a $D \times D$-matrix of ones. No direct, general transformation exists with other log-ratio covariances, although Eqs. (4.8) and (4.9) always allow the conversion to a clr-variogram and transformation from it to a variation-variogram.

These equations hold for both the population and the empirical versions. Proofs of these relations can be found in Pawlowsky-Glahn and Olea (2004), Tolosana-Delgado (2006) and Tolosana-Delgado et al. (2019). These expressions are valid

when working with alr or clr scores, by taking $\boldsymbol{\Psi}$ as specified in Eq. (2.4). This also applies to the remainder of this section.

One of the key ideas of compositional geostatistics is the equivalence between variation-variograms and *any* coordinate variogram matrix, through the equations given before. Actually, the various structural functions that can be constructed are different *representations* of the spatial dependence structure intrinsic to the compositional random function $\mathbf{Z}(\mathbf{x})$. Nevertheless it might be possible that each representation highlights certain aspects of that object while de-emphasising others. It is therefore prudent to consider several of these representations simultaneously in a structural analysis. These should include at least the variation-variograms and the coordinate variograms computed with respect to the clr-transformation. Re-computing the variograms is not required: Because of Eq. (4.10), it suffices to compute the variation-variograms, and recast them in terms of the coordinate variograms via the relevant transformation matrix $\boldsymbol{\Psi}$ (see Sect. 2.2).

Package "`gmGeostats`" provides a series of functions to convert between log-ratio coordinate variograms (as objects of class "`gstatVariogram`") and pairwise variograms (class "`logratioVariogram`"). These functions are named `as.gstatVariogram` and `as.logratioVariogram` respectively following the convention in **R**. Both need us to specify which log-ratio representation is desired:

```
> wind.alr.vg = as.gstatVariogram(wind.pwlr.vg, V = "alr")
> plot(wind.alr.vg)
```

The argument `V` can be either a string, "`alr`", "`ilr`" or "`clr`", representing the standard version of each of these transformations as provided in Chap. 2, or a matrix $\boldsymbol{\Psi}$ specifying an ad-hoc log-ratio transformation. No specific functions are available to deal with covariance functions, as these are rarely of practical use. Figure 4.9 shows an alr representation of the variation-variogram we estimated before. The same syntax as given before can be used to explore several of these representations.

4.3.6 Anisotropy

As for any random function, a regionalised composition may exhibit different behaviour in different spatial directions, a phenomenon known as *anisotropy*. A variable is called anisotropic if the spatial variability is direction-dependent, otherwise it is called isotropic. Two types of anisotropy are commonly distinguished: geometric and zonal. Any determination of the spatial characteristics of a regionalised composition has to explicitly account for isotropy or anisotropy in the calculation of the experimental variograms and in the subsequent modelling. To detect the anisotropy directions, it is useful to construct a variogram map (e.g. Fig. 4.10), which is a colour-coded or contoured plot of the variogram values (Isaaks & Srivastava, 1989) in either a Cartesian or polar coordinate system, constructed from experimental variograms calculated for many different directions. When the

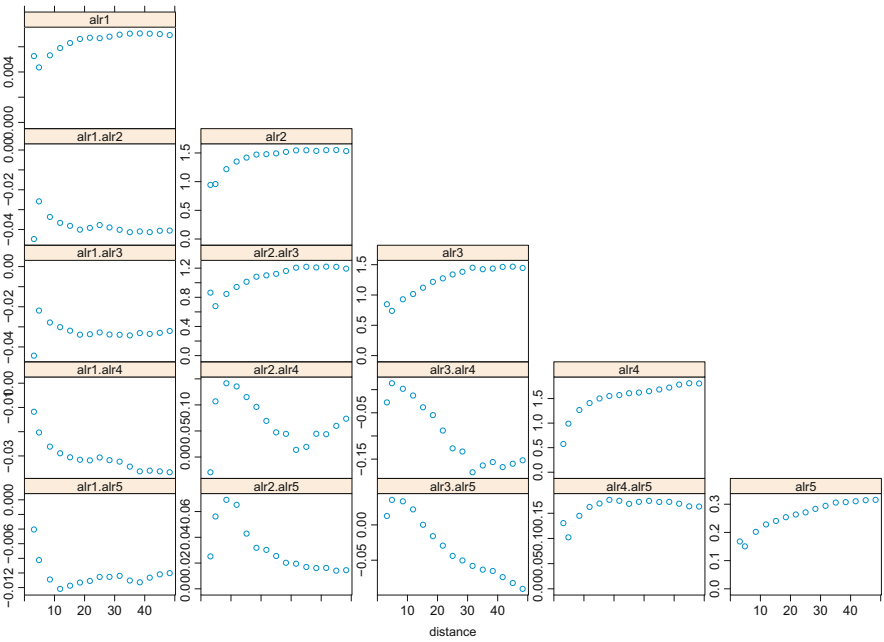

Fig. 4.9 Alr representation of the experimental direct and cross variograms of the Windarling data set

spatial variation is *isotropic*, the variogram map will show similar increases in all directions, and contours are circular. In the case of *geometric anisotropy*, the contours are elliptical and the direction of greatest continuity is the direction of the major axis of the ellipse. In the case of *zonal anisotropy*, the variogram map shows banding and the sill of the variogram depends on the direction. The direction of greatest continuity is parallel to the banding.

To decide whether or not there is evidence of anisotropy, experimental variation-variograms are computed in a variety of directions and then collated to form a *variation-variogram map*. These can then be inspected to decide whether or not anisotropy is present, and what type it represents. For the Windarling data, we will concentrate on separation distances of up to 50m and a directional variogram is calculated for every 5 degrees. To compute variation-variograms with anisotropy, the version of function `logratioVariogram` from package "`gmGeostats`" can be used

```
> wind.pwlr.vgAnis = logratioVariogram(data=acomp(wind.compo),
+                          loc=wind.coords, maxdist=50, nbins=10,
+                          azimuth.tol=15, azimuth=10*(0:35))
```

This function has the same arguments as the function `logratioVariogram` from "`compositions`": `data` and `loc` are required and respectively need to be given the compositional data and their location coordinates; arguments `maxdist`

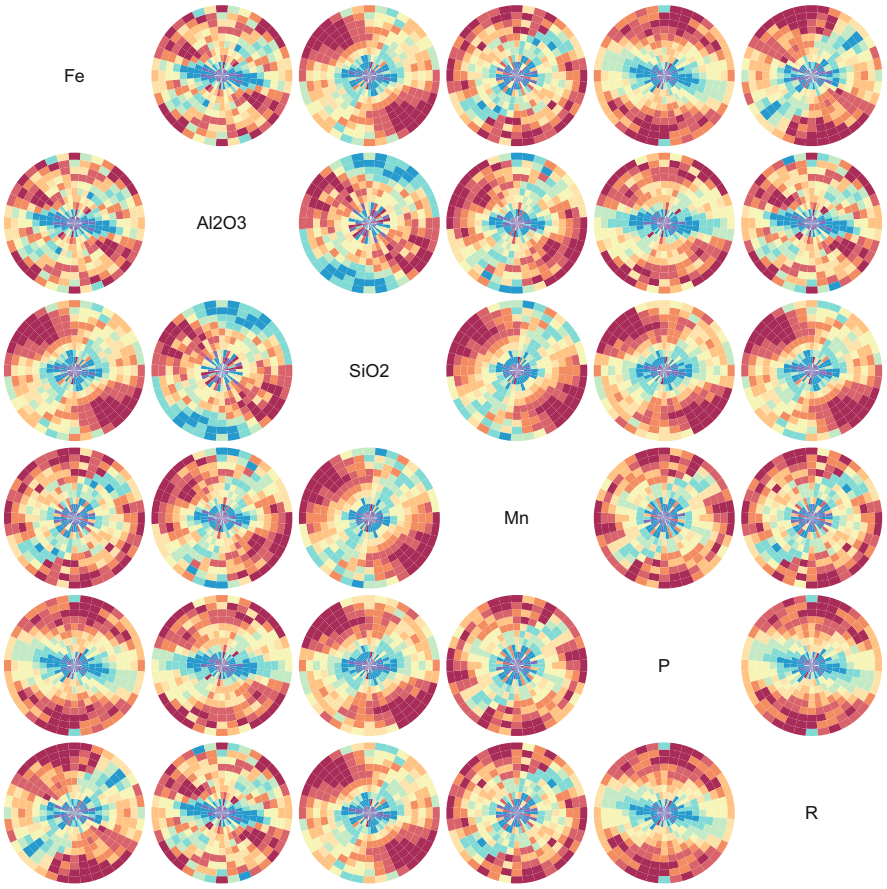

Fig. 4.10 Variation-variogram maps of Windarling data computed for a total separation distance of 50m at a nominal spacing of 5m

and `nbins` control the lag distances to be considered; finally, `azimuth` and `azimuth.tol` control the angles and their tolerances for directional variograms. The default calculations allow the generation of a variogram map (Fig. 4.10.), with blue indicating low values and red high values.

```
> image(wind.pwlr.vgAnis)
```

The majority of pairwise log-ratio variogram maps suggests greater continuity in the EW direction than in the NS direction. Consideration of the maps further suggests that the anisotropy is geometric; this is confirmed by a plot of the experimental semivariograms in directions N0 and N90.

```
> plot(wind.pwlr.vgAnis, azimuths=c("0N","90N"), lty=1)
```

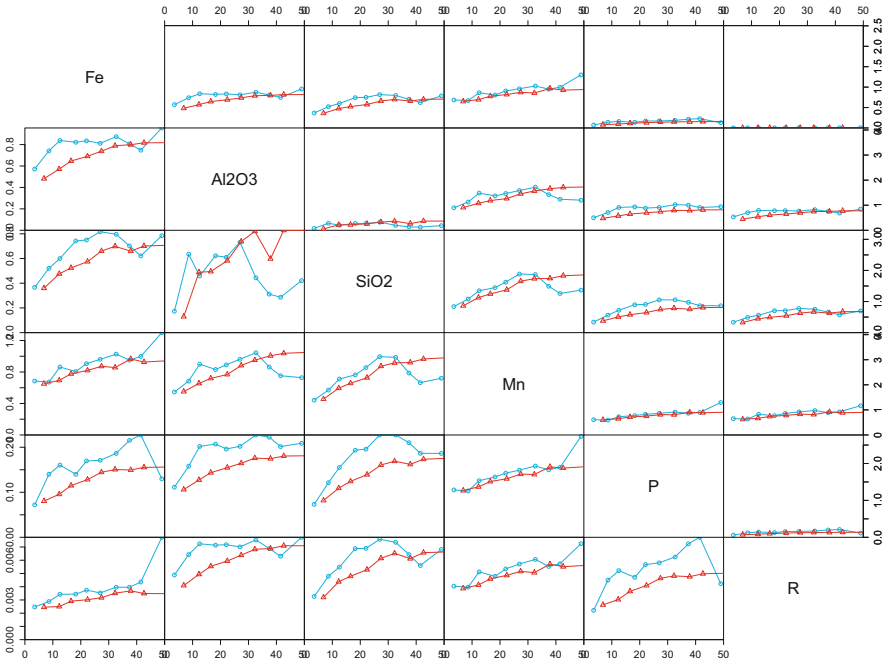

Fig. 4.11 Experimental variation-variograms of Windarling data computed for a total separation distance of 50m at a nominal spacing of 2.5m, direction N0 (NS) is shown in turquoise and direction N90 (EW) in red

Directional experimental variation-variograms are shown in Fig. 4.11. The upper plots share the same vertical scale, so that the heights of the variograms indicate which pairwise log-ratios are more variable: Fe-R, Fe-P and P-R all show very low sills, suggesting that these three variables are quite constant. The same happens with Al_2O_3-SiO_2. This supports the biplot analysis from Chap. 3. There is also a difference between the anisotropies in the East (Fig. 4.12) and West (Fig. 4.13) benches, that likely reflect the strikes of the benches.

```
> windW.pwlr.vgAnis = logratioVariogram(
+   acomp(wind.compo[!as.logical(windarling$East),]),
+   wind.coords[!as.logical(windarling$East),],
+   maxdist=45, nbins=9, azimuth.tol=15,
+   azimuth=10*(0:35) )
> image(windW.pwlr.vgAnis)

> windE.pwlr.vgAnis = logratioVariogram(
+   acomp(wind.compo[as.logical(windarling$East),]),
+   wind.coords[as.logical(windarling$East),],
+   maxdist=45, nbins=9, azimuth.tol=15,
+   azimuth=10*(0:35) )
> image(windE.pwlr.vgAnis)
```

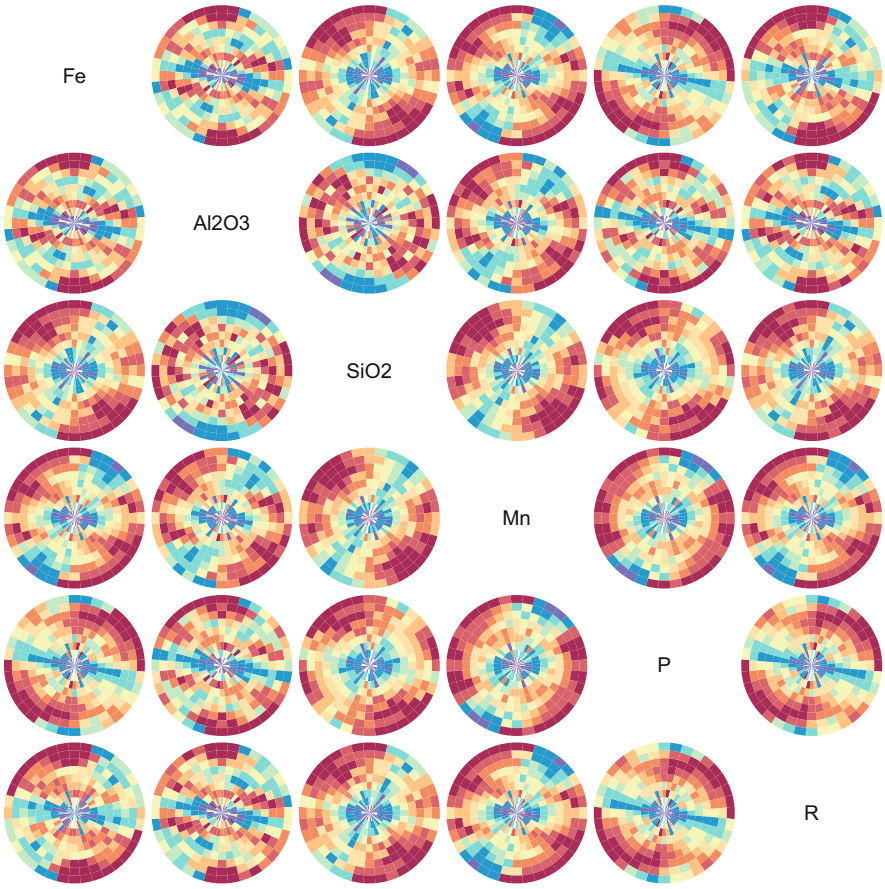

Fig. 4.12 Variation-variogram maps of the eastern part of Windarling data computed for a total separation distance of 45 m at a nominal spacing of 5m

Whether this difference is relevant or not depends on the goal of the study, and the scale further interpolations will be applied to. For example a kriging with a small neighbourhood, for instance, it might suffice to focus on short distances, where this difference is minor.

4.4 MAF

4.4.1 Method

Principal component analysis and the associated biplots were introduced in Sect. 3.3 as means for exploring relationships between variables in a given set of data. This

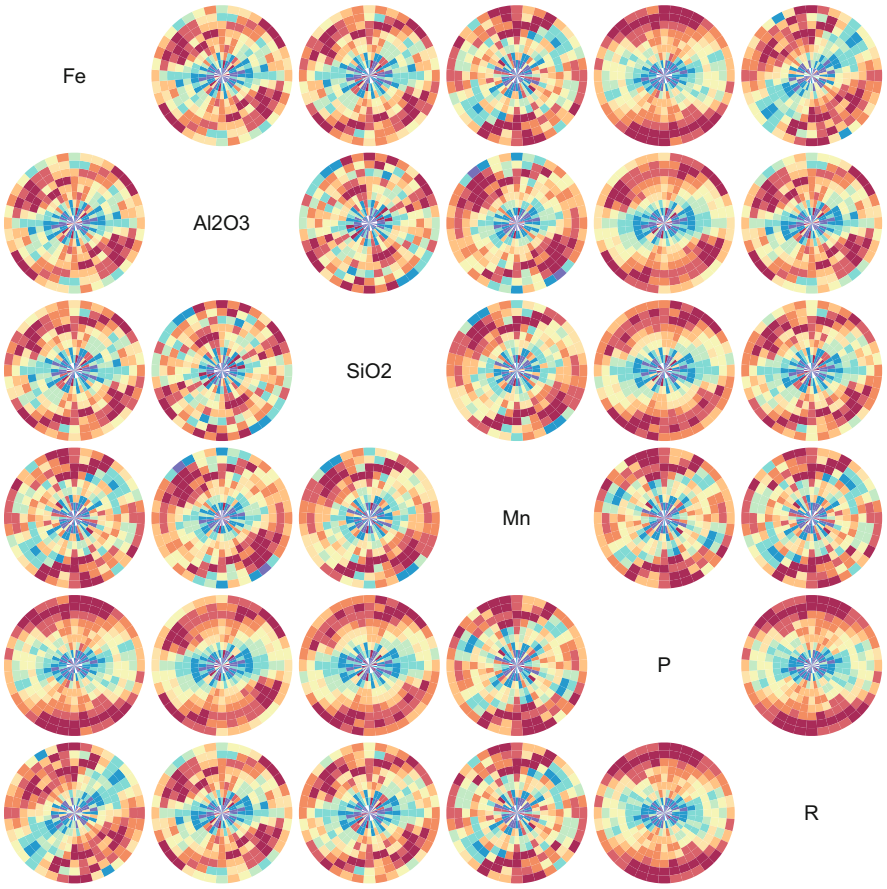

Fig. 4.13 Variation-variogram maps of western part of Windarling data computed for a total separation distance of 45 m at a nominal spacing of 5m

transformation does not account for the spatial relationships in the case of a regionalised composition, nor does it deal with the inherently noisy nature of such data. The method of Minimum/Maximum Autocorrelation factors (*MAF*) introduced by Switzer and Green (1984) was first developed to provide a mechanism for separating noise and signal in multichannel remote sensing data and makes explicit use of global spatial statistics. The method is based on the simultaneous diagonalization of the experimental covariance matrix or as a proxy a coordinate variogram matrix at sufficiently high lag and a coordinate variogram matrix that characterises the short scale variability of the data. For the case of compositional data, the calculations will be based on glr-coordinates of the regionalised composition. The transformation is typically carried out in four steps (Desbarats & Dimitrakopoulos, 2000; Bandarian et al., 2008).

1. Compute the variance–covariance matrix $\hat{\mathbf{S}}$ of the centred log-ratio transformed data $\{\boldsymbol{\zeta}(\mathbf{x}_i)\}$, determine its principal component decomposition $\hat{\mathbf{S}} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^t$ (Sect. 3.3 or Sect. A.2); remember that the entries in $\boldsymbol{\Lambda}$ are ordered by decreasing value.
2. Compute the PCA scores $\mathbf{y}^t(\mathbf{x}_i) = \boldsymbol{\zeta}^t(\mathbf{x}_i)\mathbf{Q}\boldsymbol{\Lambda}^{-1/2}$. At this stage the new variables have variance 1.
3. Compute the semivariogram matrix $\hat{\boldsymbol{\Gamma}}(\mathbf{h}_0)$ for $\mathbf{y}$ at a desired lag $\mathbf{h}_0$ and determine its spectral decomposition (Sect. A.2) again ordering the eigenvalues in descending order: $\hat{\boldsymbol{\Gamma}}(\mathbf{h}_0) = \mathbf{P}\boldsymbol{\Lambda}_1\mathbf{P}^t$.
4. Put $\mathbf{y}^t_F(\mathbf{x}_i) = \mathbf{y}^t(\mathbf{x}_i)\mathbf{P}$. Thus the overall transformation is $\mathbf{y}^t_F(\mathbf{x}_i) = \boldsymbol{\zeta}^t(\mathbf{x}_i)\mathbf{A}$ with $\mathbf{A} = \mathbf{Q}\boldsymbol{\Lambda}^{-1/2}\mathbf{P}$.

The matrix $\mathbf{A}$ diagonalises $\hat{\mathbf{S}}$ and $\hat{\boldsymbol{\Gamma}}(\mathbf{h}_0)$ simultaneously:

$$\mathbf{A}^t\hat{\mathbf{S}}\mathbf{A} = \mathbf{I}; \qquad \mathbf{A}^t\hat{\boldsymbol{\Gamma}}(\mathbf{h}_0)\mathbf{A} = \boldsymbol{\Lambda}_1.$$

In the context of compositional analysis, the matrices $\mathbf{Q}$ and $\mathbf{A}$ play the role of matrix $\boldsymbol{\Psi}$, i.e. they define the basis of the simplex used for the calculation of log-ratio coordinates. Matrix $\mathbf{Q}$ is orthogonal,[1] hence it defines an orthonormal basis; on the other hand, matrix $\mathbf{A}$ defines an oblique basis.

The MAF factors may be used to construct a representation of the original variables in terms of the factors. Command `Maf` provides a convenient way to compute the matrix $\mathbf{A}$ from the base data and the corresponding experimental variogram. These can be an "`acomp`" composition paired with a variation-variogram of class "`logratioVariogram`", or else a "`data.frame`" paired with a "`gstatVariogram`".

```
> maf.wind = Maf(acomp(wind.compo), vg=wind.pwlr.vg, i=2)
> class(maf.wind)

[1] "maf"             "genDiag"         "princomp.acomp"
[4] "princomp"

> colnames(maf.wind$scores)

[1] "maf1" "maf2" "maf3" "maf4" "maf5"
```

The outcome is a generalisation of a PCA, hence it has `scores`, `loadings` and `sdev` elements available for further use. In this way, we can easily build a "`gstat`" geostatistical object making use of the computed scores,

```
> wind.maf.gg =
+   make.gmCompositionalGaussianSpatialModel(
+     wind.compo, wind.coords, V = maf.wind$loadings,
+     prefix="maf")
```

[1] With respect to the metric underlying the starting transformed scores $\boldsymbol{\zeta} = \mathrm{glr}(\mathbf{z})$.

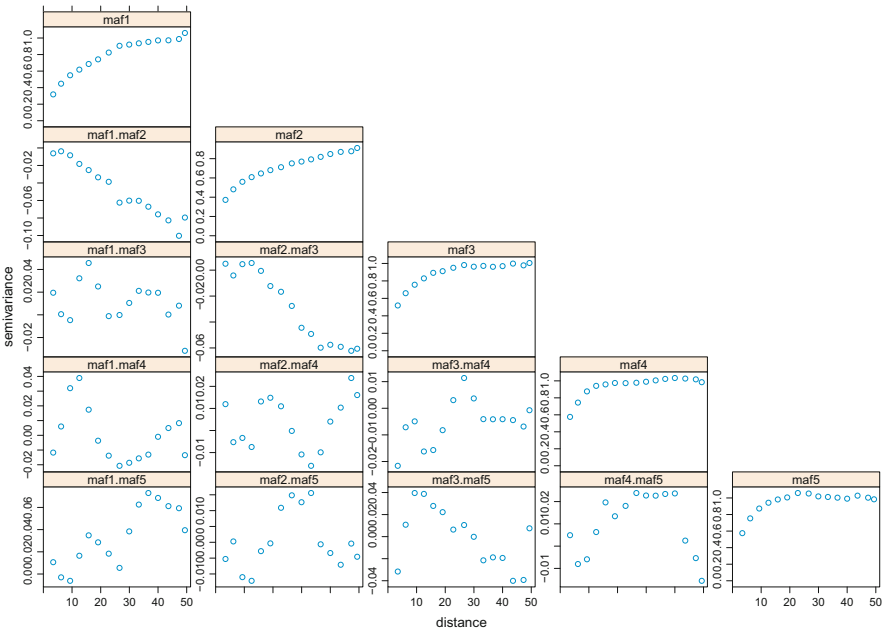

Fig. 4.14 Experimental direct and cross variograms of the Windarling MAF factors based on the ilr-covariance matrix and the matrix at lag 1

By construction the factors have the property that they are un-correlated at lags 0 and $\mathbf{h}_0$ and ordered in descending order of spatial continuity. Moreover, the correlation at other lags is usually negligible. Omnidirectional variograms and cross variograms[2] are shown in Fig. 4.14, and obtained with code such as

```
> wind.maf.vg = variogram(wind.maf.gg, cutoff=50, width=3.5)
> plot(wind.maf.vg)
```

Note that here we calculated a variogram directly from a "`gmSpatialModel`" object, without transforming it to "`gstat`" (this works as well!). The results show the behaviour discussed above in terms of the continuity of the factors, with maf1 exhibiting the greatest continuity and maf5 the least, this is evident from decreasing ranges in the direct experimental variograms and increasing nuggets. An assessment of the cross variograms is more difficult as with the gstat plotting function for variograms each cross variogram is plotted with its own vertical scale. Therefore the function `variogramModelPlot` from "`gmGeostats`" provides the option to force the variogram plots to share the vertical axes, in this case by rows

```
> variogramModelPlot(wind.maf.vg, commonAxis = T)
```

[2] If the factors do not exhibit structured spatial cross correlation, then the representation is an expansion via *empirical orthogonal functions*.

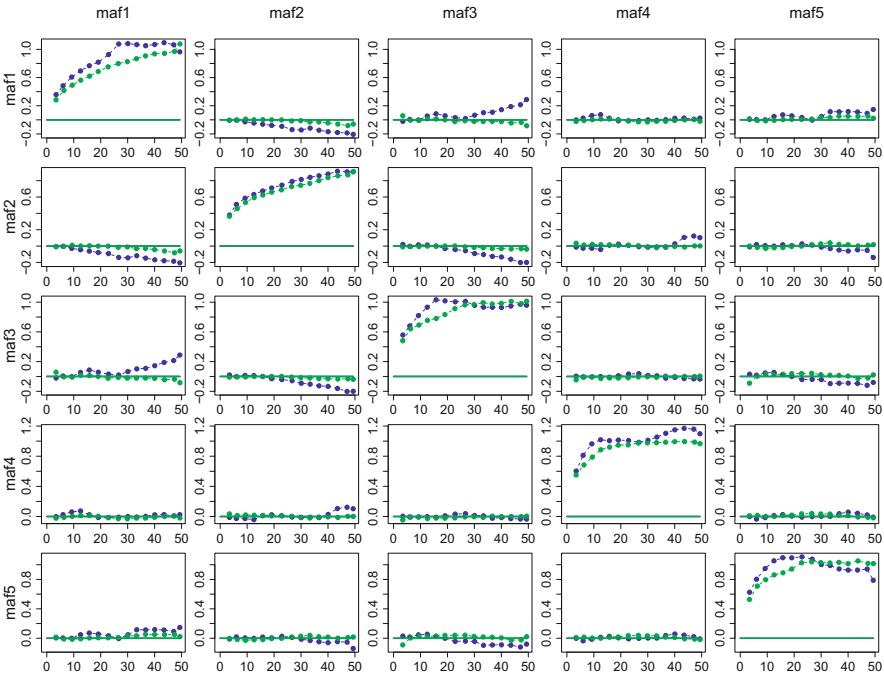

Fig. 4.15 Experimental variograms of the Windarling MAF factors based on the ilr-covariance matrix and the matrix at lag 1 in directions N90 (green) and N0 (blue)

The same applies to directional variograms of the MAF factors (Fig. 4.15)

```
> wind.maf.vgAnis = variogram(wind.maf.gg, cutoff=50,
+                     width=3.5, alpha=c(0,90))
> variogramModelPlot(wind.maf.vgAnis, commonAxis = T)
```

In this representation, it is easier to see that the ranges of direct variograms decrease and the nugget to sill ratio increases with increasing order of the factors, and that cross variograms are mostly zero at almost all lags. Of the five factors, maf1, maf3, maf4 and maf5 show clear evidence of geometric anisotropy, while maf2 exhibits isotropic behaviour. Notably, from lag distances larger than 30m on the NS direction both maf1 and maf2 factors and their cross variogram appear to show a quadratic behaviour, suggesting a NS trend. This will be further explored in subsequent chapters.

4.4.2 MAF Biplots

Just like PCA factors, the MAF factors can be used to construct biplots (Mueller et al., 2020). However, these are usually expressed in terms of clr coordinates. If $\mathbf{V}$ denotes the transformation that recasts the ilr transformed data as clr-transformed data, then a *MAF biplot* can be constructed from $\mathbf{y}^t_{MAF}(\mathbf{x}_i) = (\boldsymbol{\zeta}^t_{clr}(\mathbf{x}_i) - \bar{\boldsymbol{\zeta}}^t_{clr})\mathbf{V}\mathbf{Q}\boldsymbol{\Lambda}^{-1/2}\mathbf{P}$ where $\bar{\boldsymbol{\xi}}_{clr}$ denotes the mean of the clr coordinates. A check of the MAF biplot (Fig. 4.16) indicates a completely different association structure among the variables, in particular between Fe and R:

```
> par(mfrow=c(1,2))
> coloredBiplot(maf.wind, xlabs.pc=19, xlabs.col=
+                 as.integer(windarling$Lithotype),
+                 cex=c(0.5,1), col="black")
> coloredBiplot(maf.wind, xlabs.pc=19, xlabs.col=
+                 as.integer(windarling$Lithotype),
+                 cex=c(0.5,1), col="black", choice=c(3,4))
```

It is interesting to see that, while the ratios Fe/R, Fe/P or R/P are highlighted to be almost constant in the non-spatial PCA, here Fe/R has no contribution whatsoever onto the first two MAF factors (only P contributes to MAF factor 1 notably). This first maf factor actually highlights a subcomposition SiO_2-Al_2O_3-P, whereas the second factor is showing a strong relation with Mn/P. Note that these two factors are those which exhibit a certain NS trend, the longest range and the lowest nugget in Fig. 4.15. Figure 4.16 shows as well that one needs to consider factor maf4 to observe some structure between other variables, in this case within the subcomposition (SiO_2, Mn, Al_2O_3), with a moderate range between 20m and 30m and 40% nugget/sill ratio. It is also on this biplot that we see Fe and R variations, namely in the fourth component! This implies that Fe/R variations have very short

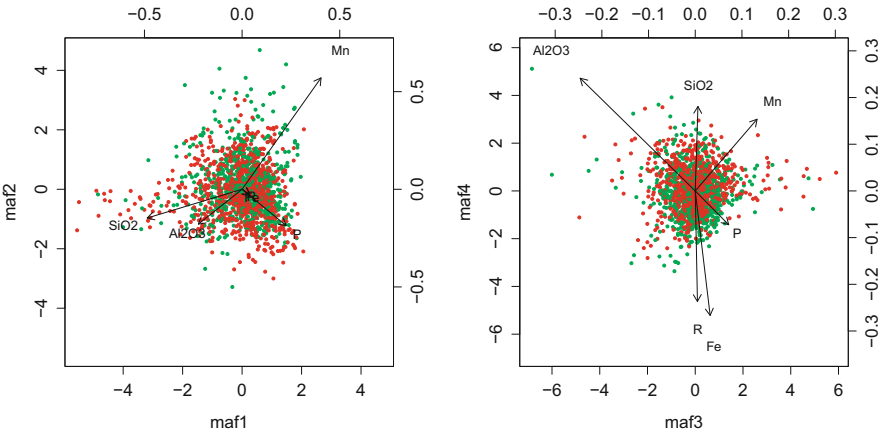

Fig. 4.16 MAF biplots of first vs. second factor and of third vs. fourth factor, based on the covariance matrix and the matrix at lag 1

continuity, i.e. Fe/R variability occurs at a very small spatial scale, which is not altogether inconsistent. The joint exploration of these biplots and the variography structure of their associated factors can indeed deliver rich information about the spatial scales dominating on each subcomposition.

In addition to the construction and exploratory analysis of spatially aware biplots, MAF can be used for dimension reduction by means of the expansion discussed earlier on (Petitgas et al., 2018). This reduction can be based on a sequence of approximations to the data based on the first k factors, $k = 1, \ldots, D$ and that approximation is chosen where the total variance explained exceeds a particular threshold. Further, in the case where the factors form a family of empirical orthogonal functions, the factors can be modelled separately and estimates or simulations derived from them.

4.5 Checks of Spatial Structure

4.5.1 Spatial Decorrelation

To check whether or not the MAF transformed data are adequately spatially decorrelated, the plot of the experimental direct semivariograms and cross variograms is inspected to assess lack of structure in the cross variograms and their closeness to 0. In the case of the Windarling data (Fig. 4.15), the cross variograms of the MAFs have negligible sills and so the factors can be deemed to form a family of empirical orthogonal functions. This criterion is quite subjective, so that quantitative measures are also considered. The spatial decorrelation of the factors can be assessed via the quantitative measures introduced in Tercan (1999). These are

- The *absolute deviation from diagonality* $\zeta(\mathbf{h})$, defined as the sum of squares of the off-diagonal elements of the factor experimental semivariogram matrix at lag $\mathbf{h}$:

$$\zeta(\mathbf{h}) = \sum_{k=1}^{n} \sum_{j=1, j\neq k}^{n} \gamma_{k,j}^2(\mathbf{h}). \tag{4.12}$$

- The *relative deviation from diagonality* $\tau(\mathbf{h})$ which compares the absolute sum of off-diagonal elements with the sum of the absolute values of the diagonal elements of the factor experimental semivariogram matrix $\boldsymbol{\Gamma}_{MAF}(\mathbf{h})$) for each lag $\mathbf{h}$:

$$\tau(\mathbf{h}) = \frac{\sum_{k=1}^{n} \sum_{j=1, j\neq k}^{n} |\gamma_{k,j}(\mathbf{h})|}{\sum_{k=1}^{n} |\gamma_{k,k}(\mathbf{h})|}. \tag{4.13}$$

- The *spatial diagonalisation efficiency* $\kappa(\mathbf{h})$ which compares the sum of squares of off-diagonal elements in $\boldsymbol{\Gamma}_{\mathbf{F}}(\mathbf{h})$) with those of the attribute semivariogram matrix $\boldsymbol{\Gamma}_{\mathbf{Z}}(\mathbf{h})$)

$$\kappa(\mathbf{h}) = 1 - \frac{\sum_{k=1}^{n} \sum_{j=1, j \neq k}^{n} \gamma_{k,j}^2(\mathbf{h})}{\sum_{k=1}^{n} \sum_{j=1, j \neq k}^{n} \gamma_{(0)k,j}^2(\mathbf{h})} \tag{4.14}$$

Here $\gamma_{k,j}(.)$ denotes the experimental cross-semivariogram for the k-th and j-th coordinates of the MAF transformed data $\{\mathbf{y}_F(\mathbf{x}_i)\}$, and $\gamma_{(0),k,j}(.)$ is the experimental cross-semivariogram for the k-th and j-th log-ratio coordinates of $\mathbf{Z}(\mathbf{x})$. The number of variables n will typically be $D-1$, the number of MAF factors.

Function `spatialDecorrelation` computes each of these decorrelation measures, depending on an extra argument `method`, which can be a combination of "`add`", "`rdd`" or "`sde`", respectively for $\zeta(\mathbf{h})$, $\tau(\mathbf{h})$ or $\kappa(\mathbf{h})$:

```
> aux = spatialDecorrelation(wind.maf.vg, method=c("add", "rdd"))
> summary(aux)
```

```
       np              dist           gamma            dir.hor
 Min.   : 7220   Min.   : 3.4   Min.   :0.0003   Min.   :0
 1st Qu.:22011   1st Qu.:13.4   1st Qu.:0.0034   1st Qu.:0
 Median :42336   Median :26.6   Median :0.0062   Median :0
 Mean   :33611   Mean   :26.4   Mean   :0.0107   Mean   :0
 3rd Qu.:44278   3rd Qu.:39.2   3rd Qu.:0.0121   3rd Qu.:0
 Max.   :49592   Max.   :49.4   Max.   :0.0395   Max.   :0
    dir.ver    id
 Min.   :0   add:15
 1st Qu.:0   rdd:15
 Median :0
 Mean   :0
 3rd Qu.:0
 Max.   :0
```

```
> class(aux)
```

```
[1] "spatialDecorrelationMeasure" "gstatVariogram"
[3] "data.frame"
```

The result has a structure inherited from the "`gstat`" variogram class, so that many post-processing functions (e.g. `plot`) work directly. Figure 4.17 shows the three diagonalisation efficiency measures, for the original Windarling alr-transformed composition and the MAF transformation. For a "good" decorrelation, values for $\zeta(\mathbf{h})$ and $\tau(\mathbf{h})$ should be near zero, and $\kappa(\mathbf{h})$ near one.

The averages of the three measures over the set of lags will be denoted by $\bar{\zeta}$, $\bar{\tau}$ and $\bar{\kappa}$, respectively. Ideal values for them are again 0 for the first two, and 1 for $\bar{\kappa}$. No function exists to compute them, but they just require a simple calculation

```
> aux = spatialDecorrelation(wind.maf.vg, method="add")
> mean(aux)
```

```
   add
0.0172
```

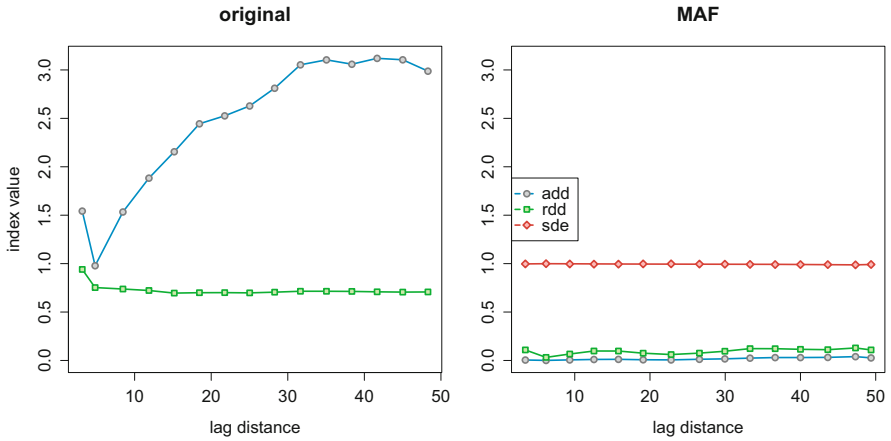

Fig. 4.17 Spatial diagonalisation measures for the original alr-transformed Windarling data set, and for its MAF representation

```
> aux = spatialDecorrelation(wind.maf.vg, method="rdd")
> mean(aux)

   rdd
0.0945

> aux = spatialDecorrelation(wind.maf.vg, wind.alr.vg,
+                            method="sde")
> mean(aux)

  sde
0.994

> plot(aux)
> spatialDecorrelation(wind.maf.vg, method="rdd")
> spatialDecorrelation(wind.maf.vg, wind.alr.vg, method="sde")
```

4.5.2 Spatial Independence

An aspect that has not yet been discussed is the possibility of the spatial dependence being the result of unstructured stochastic variations, in other words, that the data might be spatially independent. This can be tested with a permutation test. Inspired by the MAF decomposition, the following test is implemented here in the function `noSpatCorr.test`.

```
> noSpatCorr.test(acomp(wind.compo), wind.coords,
+                 maxlag0=3, minlagInf=8)
```

The idea is to compare two matrices, one characterising the long-range and one the short-range covariance of increments, and to measure how similar these two are. In

this case we would do the calculations taking for the short-range covariance all pairs of points less than $3m$ apart, and for the long-range the pairs of points more than $8m$ apart.

Let $\mathbf{C}_0$ and $\mathbf{C}_1$ respectively be the short- and long-range covariance matrices. Using a matrix $\mathbf{R}_1$ such that $\mathbf{C}_1 = \mathbf{R}_1 \cdot \mathbf{R}_1^t$ (calculated e.g. with the SVD as explained in Sect. A.4), the proposed test statistic is the smallest eigenvalue λ_0 (see Sect. A.2) of the matrix

$$\mathbf{W} = \mathbf{R}_1^{-t} \mathbf{C}_0 \mathbf{R}_1^{-1}.$$

Under the null hypothesis of lack of spatial correlation, $\mathbf{W} \approx \mathbf{I}$, and $\lambda_0 \approx 1$. The distribution of λ_0 can be approximated by computing several times (default 299 times) a permutation of the rows of the coordinate data set (effectively destroying spatial correlation while keeping all other characteristics of the composition intact) and producing the test statistic for each of these. Comparing λ_0 for the true data set with this sample, one can assess how credible the null hypothesis is.

4.6 Example: Tellus

The Tellus data set of Northern Ireland (Sect. 1.3.2) contains 6862 samples. To ease its use as an illustration tool, we have defined a reference subset, controlled by the variable `Flag`. In this section, we will analyse the Tellus data set once in its complete form, and once for this reference subset only

```
> setwd("YOUR_WORKING_DIRECTORY")
> getTellus(cleanup = TRUE, TI = FALSE)
> load("TellusASoil.RData")
> tellus=TellusASoil

> dim(tellus)

[1] 6862   58

> colnames(tellus)

 [1] "Sample"   "EASTING"  "NORTHING" "Flag"     "Ag"
 [6] "Cd"       "In"       "Sn"       "Sb"       "Te"
[11] "I"        "Cs"       "Ba"       "La"       "Ce"
[16] "Na2O"     "MgO"      "Al2O3"    "SiO2"     "P2O5"
[21] "SO3"      "K2O"      "CaO"      "TiO2"     "MnO"
[26] "Fe2O3"    "Cl"       "Sc"       "V"        "Cr"
[31] "Co"       "Ni"       "Cu"       "Zn"       "Ga"
[36] "Ge"       "As"       "Se"       "Br"       "Rb"
[41] "Sr"       "Y"        "Zr"       "Nb"       "Mo"
[46] "Nd"       "Sm"       "Yb"       "Hf"       "Ta"
[51] "W"        "Tl"       "Pb"       "Bi"       "Th"
[56] "U"        "pH"       "LOI"
```

```
> tellus.coords = tellus %>% dplyr::select(EASTING, NORTHING)
> tellus.compo0 = tellus %>%
+   dplyr::select(Na2O:P2O5, K2O:Fe2O3, LOI)
> tellus.compo = tellus %>%
+   dplyr::select(MgO, Al2O3, CaO, Fe2O3) %>%
+   mutate(R=100-MgO-Al2O3-CaO-Fe2O3)
> tellus.subset = as.logical(tellus$Flag)
> summary(tellus.subset)

   Mode    FALSE     TRUE
logical     6497      365
```

Now `tellus.coords` contains the spatial coordinates of all points, `tellus.compo0` the major oxides, `tellus.compo` four chosen major oxides plus the rest R to ensure the sum to 100%, and `tellus.subset` is a Boolean vector determining which data rows belong to the reference subset. Due to space reasons, most of the following analyses are done with the subcomposition of MgO, Al_2O_3, CaO, Fe_2O_3 and R; readers are invited to repeat them for the entire major oxide composition.

With these objects we first study the variograms of the subset. Variation-variograms are displayed in Fig. 4.18.

```
> tellusS.pwlr.vg = logratioVariogram(
+                   acomp(tellus.compo[tellus.subset,]),
```

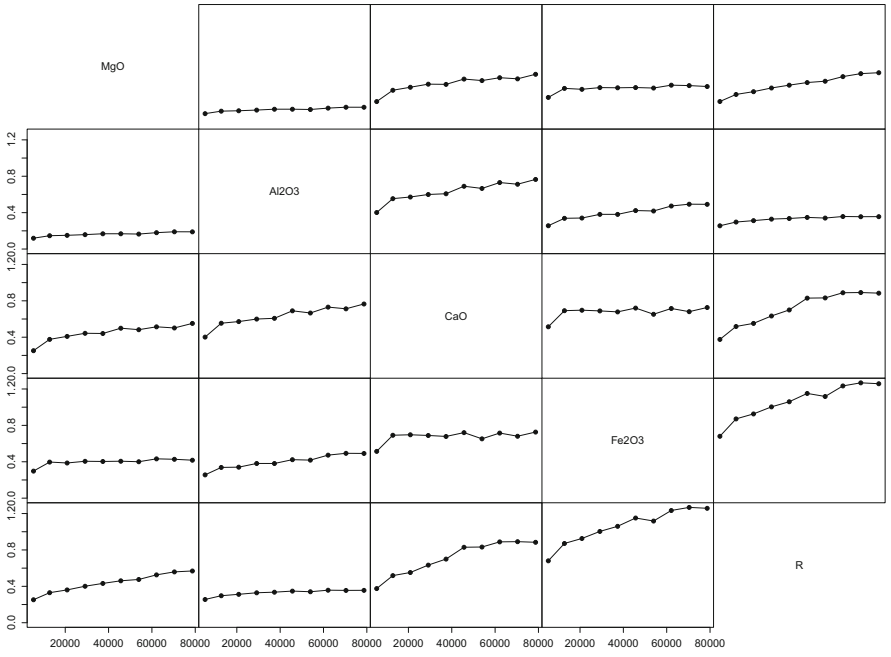

Fig. 4.18 Experimental variation-variograms obtained for the reference subset of Tellus

```
+                     tellus.coords[tellus.subset,], nbins=10)
> plot(tellusS.pwlr.vg, type="o", pch=19)
```

These can then be represented in several log-ratio coordinate systems, in order to understand their structure.

```
> tellusS.alr.vg = as.gstatVariogram(tellusS.pwlr.vg, V = "alr")
> tellusS.ilr.vg = as.gstatVariogram(tellusS.pwlr.vg, V = "ilr")
> plot(tellusS.ilr.vg)
```

Here we include an image of the ilr variograms only (Fig. 4.19). Variograms in alr-coordinates will be shown in Chap. 5, where this example will be discussed further.

We continue the analysis of the spatial structure via PCA and MAF. A biplot from a principal component analysis of the subset data set for the whole major oxides composition can be obtained with

```
> tellus.compo0[tellus.subset,] %>%  acomp %>% princomp %>%
+   biplot(xlabs=rep(".", sum(tellus.subset)))
```

The biplot (Fig. 4.20, upper left panel) shows a sort of three-axis structure, with LOI, MnO and K_2O on the extremes. We see as well a series of triplets of variables which are aligned, suggesting possible ternary diagrams with a one-dimensional pattern, for instance (Fe_2O_3, CaO, LOI). This code would produce a plot of this subcomposition with different colours and symbols for the reference set (red circles)

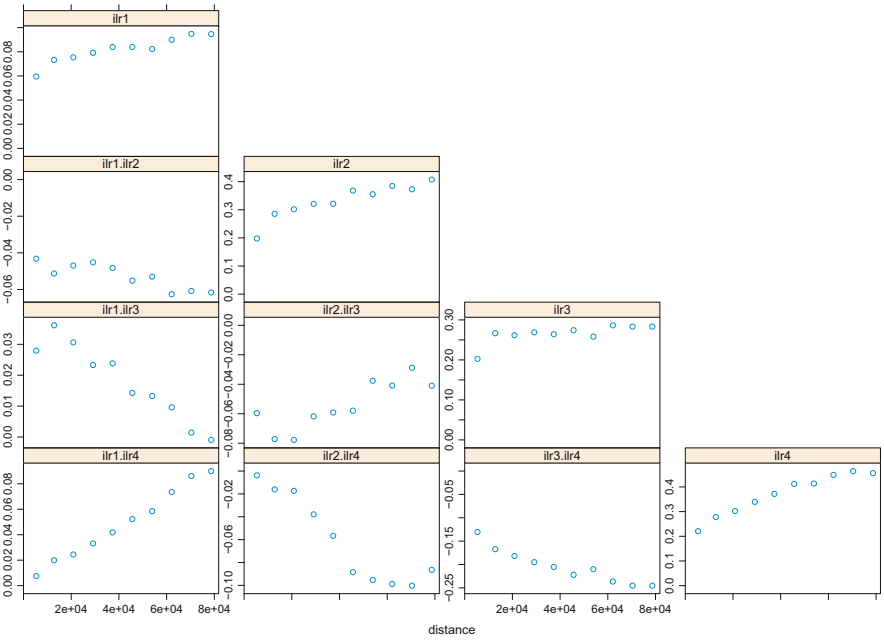

Fig. 4.19 ilr variograms obtained for the reference subset of Tellus

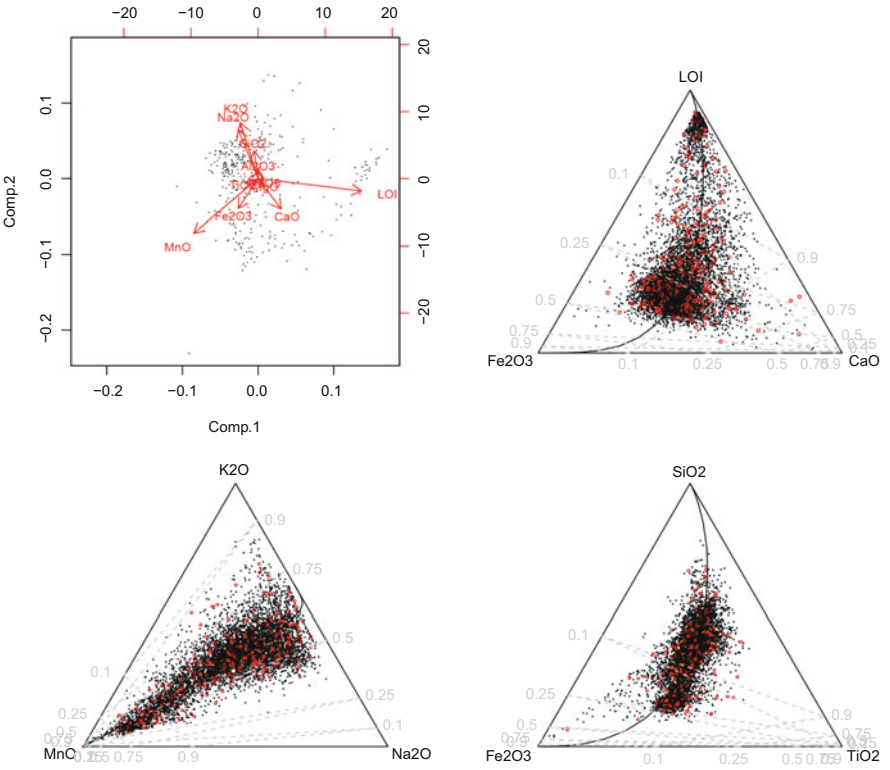

Fig. 4.20 Biplots and selected ternary diagrams of the Tellus data set. The biplot is obtained for the reference subset. The ternary diagrams show both the reference subset (red circles) and the complete set (black dots)

and the complete set (black points), and a reference grid

```
> tellus.compo0 %>% dplyr::select(Fe2O3, CaO, LOI) %>% acomp %>%
+   plot(cex=0.35*tellus.subset+0.1, col=tellus.subset+1,
+        pca=T, center=T)
> isoPortionLines(at=c(0.1, 0.25, 0.5, 0.75, 0.9), col="grey",
+                 lty=2)
```

This and the subcompositions (MnO, K_2O, Na_2O) and (Fe_2O_3, SiO_2, TiO_2) are displayed in Fig. 4.20 as well. To compute the MAF, we need to provide a data set, its empirical variogram and the index of the lag distance of interest,

```
> maf.tellusS = Maf(acomp(tellus.compo[tellus.subset,]),
+                           vg = tellusS.pwlr.vg, i=2)
```

Figure 4.21 shows the biplot of this MAF, together with the MAF of the whole data set. It can be seen that the structure obtained for the subset is very similar to the one obtained for the whole data set, with a mirroring of the vertical axis (MAF and

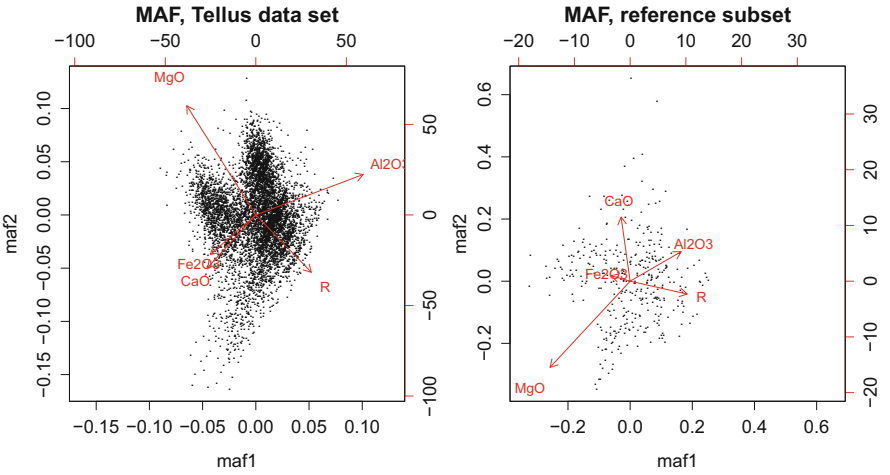

Fig. 4.21 MAF biplot of the Tellus data set, for the whole data (left) and for the reference subset (right)

PCA scores can show sign reversals). These biplots show that the subset is, from the perspective of a MAF calculation, representative of the whole data set.

We finally compute the variograms of the MAF transformed subcomposition, which requires firstly the creation of a spatial data container object,

```
> tellusS.maf.gg = make.gmCompositionalGaussianSpatialModel(
+       acomp(tellus.compo[tellus.subset,]),
+       tellus.coords[tellus.subset,],
+       V = maf.tellusS$loadings,prefix = "maf")
> tellusS.maf.vg = variogram(tellusS.maf.gg,
+                       boundaries=tellusS.ilr.vg$dist[1:10])
> variogramModelPlot(tellusS.maf.vg, commonAxis = T, col=2)
```

Figure 4.22 shows the resulting variogram plot. Figure 4.23 displays the spatial decorrelation measures of the alr, the ilr and the MAF representations of the data set, all obtained with calls to `spatialDecorrelation` as shown in Sect. 4.5.1.

Finally, we can consider the possibility that the data from the reference subset do not contain sufficient statistical evidence of spatial correlation

```
> noSpatCorr.test(tellusS.maf.gg, maxlag0=20e3, minlagInf=50e3)

        permutation penalized eigenvalue ratio

data:
= 0.5, p-value <2e-16
```

In this case, we used the test by just specifying the "`gmSpatialModel`" object, which also works. The result is a p-value around zero for the null hypothesis of lack of spatial autocorrelation, which allows us to conclude that the data are indeed sufficiently autocorrelated to continue with a spatial analysis.

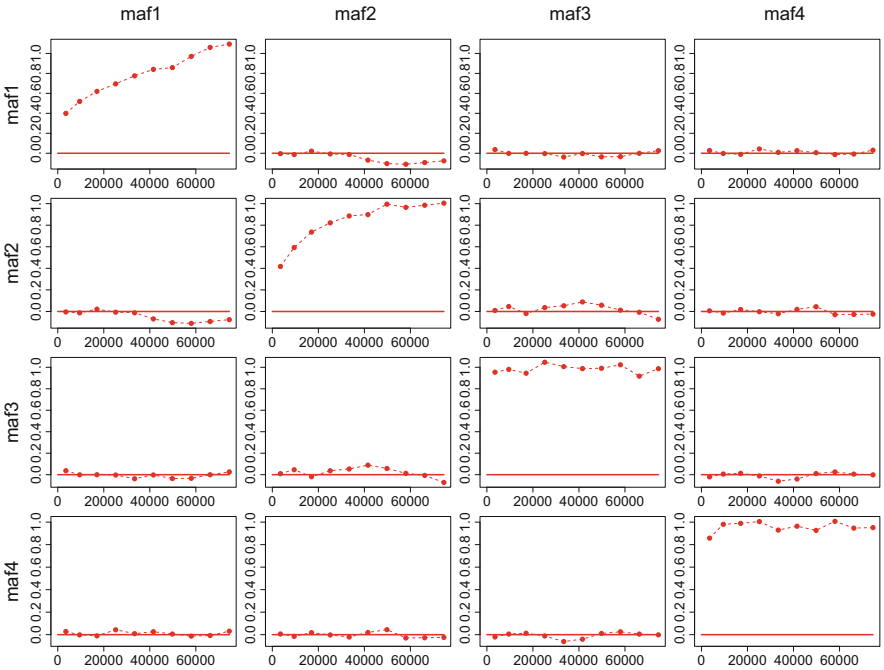

Fig. 4.22 MAF variograms of the Tellus data set, reference subset, composition (MgO, Al_2O_3, CaO, Fe_2O_3, R)

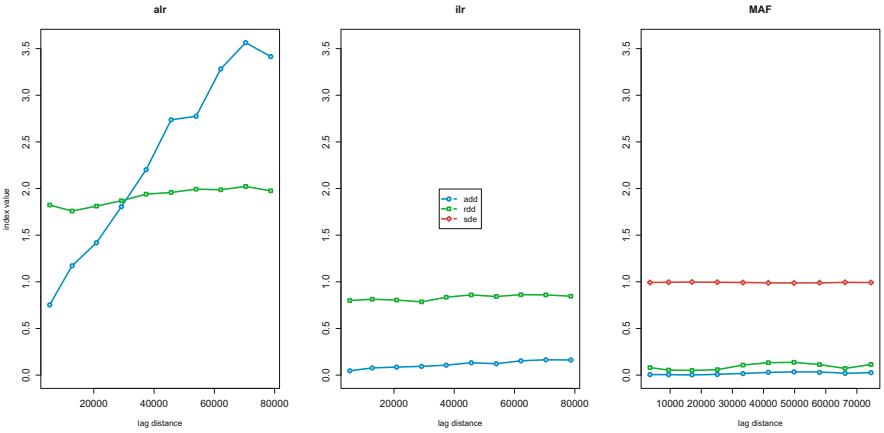

Fig. 4.23 Graphs of the spatial decorrelation measures for alr-transformed (left), ilr-transformed (centre) and MAF transformed (right) Tellus subcomposition

Problems

4.1 Variography of the Tellus Oxides

(a) For the major oxides within the Tellus data set, construct pairwise log-ratio variograms and log-ratio variogram maps and hence discuss the spatial continuity of these subcompositions.
(b) Construct PCA and MAF biplots for this composition and also check the dependence of the MAF biplot on the choice of lag.

4.2 Isotropic Variography of Major Elements in the NGSA Data Set

(a) Consider the NGSA data set, filtering it to keep only those observations from `REGION=="EAST"` and `CODE=="Bc"` (bottom soil, coarse fraction). Evaluate the minimum and maximum geographic distance between samples, and choose appropriate parameters (maximum lag, number of lags) for variogram estimation.
(b) Select the major components of the NGSA data set, those measured with XRF (thus having that string on the column name). Replace the values below detection limit by the detection limit itself. Estimate the log-ratio variograms.
(c) Construct PCA and MAF biplots for this composition.
(d) Plot experimental variograms of the principal components; plot variograms of the MAF factors.
(e) Calculate the absolute deviation, relative deviation from diagonality and the spatial diagonalisation efficiency for both PCA and MAF factors. Which of the two factorisations results in the better spatial decorrelation?

4.3 Anisotropic Variography of the Major Elements in the NGSA Data Set

(a) Consider the NGSA data set, filtering it to keep only those observations from `REGION=="EAST"` and `CODE=="Bc"` (bottom soil, coarse fraction). Evaluate the minimum and maximum geographic distance between samples, and choose appropriate parameters (maximum lag, number of lags, azimuths, azimuth tolerance) for anisotropic variogram estimation.
(b) Select the major components of the NGSA data set, those measure with XRF (thus having that string on the column name). Replace the values below detection limit by the detection limit itself. Estimate the anisotropic log-ratio variograms and plot them with `image`. If necessary, increase the azimuth tolerance. Do you see any consistent anisotropy? Does any pair of variables show strong destructurisation?
(c) Repeat for other subsets as defined by `CODE`.

References

Bandarian, E. M., Bloom, L. M., & Mueller, U. A. (2008). Direct minimum/maximum autocorrelation factors within the framework of a two structure linear model of coregionalisation. *Computers & Geosciences, 34*(3), 190–200.

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied spatial data analysis with R, 2nd edition* (405 pp.). New York: Springer Verlag.

Chilès, J. P., & Delfiner, P. (2012). *Geostatistics - Modeling spatial uncertainty* (2 ed., 699 pp.). Hoboken, NJ, USA: Wiley.

Desbarats, J., & Dimitrakopoulos, R. (2000). Geostatistical simulation of regionalised pore-size distributions using min/max autocorrelation factors. *Mathematical Geology, 32*(8), 919–942.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation* (483 pp.). Applied Geostatistics Series. New York, NY, USA: Oxford University Press.

Isaaks, E. H., & Srivastava, R. M. (1989). *An introduction to applied geostatistics* (561 pp.). New York, NY, USA: Oxford University Press.

Mueller, U., Tolosana-Delgado, R., Grunsky, E. C., & McKinley, J. M. (2020). Biplots for compositional data derived from generalized joint diagonalization methods. *Applied Computing and Geosciences, 8*, 100044.

Pawlowsky, V. (1986). *Räumliche Strukturanalyse und Schätzung ortsabhängiger Kompositionen mit Anwendungsbeispielen aus der Geologie* (170 p.). Ph.D. thesis, Fachbereich Geowissenschaften, Freie Universität Berlin, Berlin (D).

Pawlowsky-Glahn, V., & Olea, R. A. (2004). *Geostatistical analysis of compositional data*. In DeGraffenreid, Jo Anne (Ed.) Number 7 in Studies in Mathematical Geology. Oxford University Press.

Petitgas, P., Woillez, M., Doray, M., & Rivoirard, J. (2018). Indicator-based geostatistical models for mapping fish survey data. *Mathematical Geosciences, 50*(2), 187–208.

Switzer, P., & Green, A. A. (1984). Min/Max autocorrelation factors for multivariate spatial imaging. Technical Report 6, Department of Statistics, Stanford University.

Tercan, A. (1999). Importance of orthogonalization algorithm in modelling conditional distributions by orthogonal transformed indicator methods. *Mathematical Geology*, *31*(2), 155–174.

Tolosana-Delgado, R. (2006). *Geostatistics for constrained variables: positive data, compositions and probabilities. Application to environmental hazard monitoring* (198 p.). Ph.D. thesis, Universitat de Girona (Spain).

Tolosana-Delgado, R., Boogaart, K. G. v. d., & Pawlowsky-Glahn, V. (2011). Geostatistics for compositions. In V. Pawlowsky-Glahn, A. Buccianti (Eds.), *Compositional data analysis: Theory and applications* (pp. 73–86, 378 pp.). John Wiley & Sons.

Tolosana-Delgado, R., Mueller, U., & Boogaart, K. (2019). Geostatistics for compositional data: an overview. *Mathematical Geosciences, 51*(4), 485–526.

Chapter 5
Variogram Models

Abstract Here we look at model fitting. The structural functions mainly used for model fitting are introduced. The main tool for model fitting is the linear model of coregionalisation, but the application of the MAF transformation to build a linear model of coregionalisation is also demonstrated.

5.1 Linear Model of Regionalisation

In the previous chapter we defined the population and experimental versions of the variogram. In order to be able to conduct estimation and simulation, a model of the spatial continuity is required that does not depend on sample spacing, but provides the ability of measuring dissimilarity at any scale. In the univariate case this is achieved by fitting a *linear model of regionalisation* (*LMR*). The idea behind this concept is that the given random functions can be written as a linear combination of independent random functions that describe the behaviour at distinct spatial scales. In the case when the random function is second order stationary, the random function is written as

$$Z(\mathbf{x}) = \sum_{i=1}^{p} Y_i(\mathbf{x}) + m, \tag{5.1}$$

where $\text{cov}(Y_i(\mathbf{x}), Y_j(\mathbf{x}+\mathbf{h})) = 0$ for $i \neq j$, $\text{var}[Y_i(\mathbf{x})] = a_i^2$ and $m = E[Z(\mathbf{x})]$. In the intrinsic case no assumption about the mean is made and

$$Z(\mathbf{x}) = \sum_{i=1}^{p} Y_i(\mathbf{x}), \tag{5.2}$$

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_5

where the random functions Y_i are intrinsically second order stationary, with $\text{cov}(Y_i(\mathbf{x}), Y_j(\mathbf{x}+\mathbf{h})) = 0$. The semivariogram of the random function can then be described as the sum of variograms γ_i of the component random functions $Y_i(\mathbf{x})$ (Wackernagel, 2003)

$$\gamma_Z(\mathbf{x}) = \sum_{i=1}^{p} \gamma_i(\mathbf{x}). \tag{5.3}$$

In order to be able to evaluate the variogram at each separation distance a model needs to be fitted, based on the underlying experimental data. This model is derived from the LMR and is usually stated as

$$\gamma_Z(\mathbf{x}) = \sum_{i=1}^{p} a_i^2 g_i(\mathbf{h}) \tag{5.4}$$

and in order to guarantee the positivity of the kriging variance (discussed in the next chapter), the choice of functions g_i is restricted to a list of functions that are conditionally negative definite (Wackernagel, 2003). The most commonly used *variogram models* are given in Table 5.1 and represented in Fig. 5.1. When reporting the parameters of a fitted model the first model is the *nugget effect* and by convention the remaining models are listed in ascending order of spatial scale.

Table 5.1 Most commonly used variogram models, expressed as functions of an anisotropic dimensionless distance $r^2 = \mathbf{h}^t\mathbf{E}^{-1}\mathbf{h}$, where $\mathbf{E}$ describes an ellipse (or ellipsoid in 3D space) with non-negligible spatial correlation, $\delta(r) = 1$ for $r = 0$ and $\delta(r) = 0$ for $r > 0$, and χ_1 denotes the characteristic function for the set $\{r : 0 \le r \le 1\}$

Shape	Name	Normalised formula
–	Nugget	$1 - \delta(h)$
–	Spherical	$(3r/2 + r^3/2)\chi_1(r) + (1 - \delta(r))(1 - \chi_1(r))$
–	Exponential	$1 - \exp(-r/3)$
–	Gaussian	$1 - \exp(-r^2/3)$
$0 < \nu < 2$	Generalised linear	r^ν
$0 < \nu < 2$	Stable	$\exp(-r^\nu/3)$
–	Cardinal sine	$1 - \sin(r)/r$

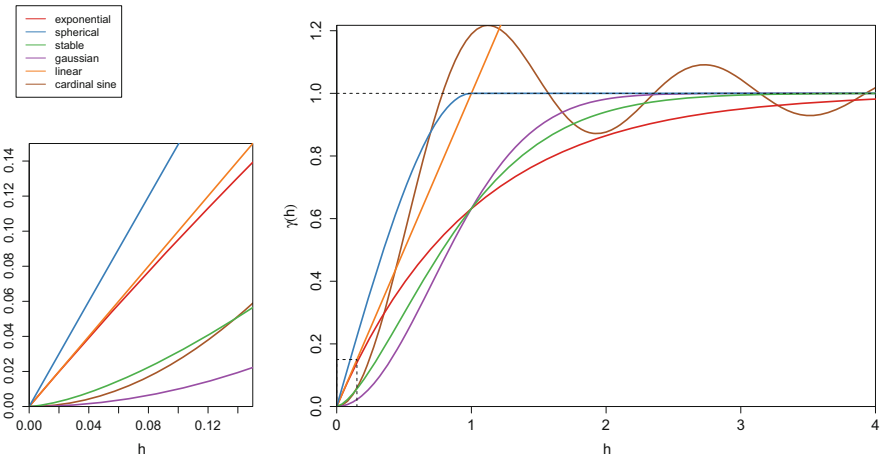

Fig. 5.1 Most commonly used unitary variogram models, behaviour near the origin is shown in the left plot

5.2 Linear Model of Coregionalisation

5.2.1 Multivariate Random Function

In analogy with the LMR, a multivariate random function is modelled as a linear combination of multivariate structures

$$\mathbf{Z}(\mathbf{x}) = \sum_{k=0}^{K} \mathbf{Y}_k + \mathbf{m},$$

where $\text{cov}(\mathbf{Y}_i(\mathbf{x}), \mathbf{Y}_j(\mathbf{x}+\mathbf{h})) = \mathbf{0}$ for $i \neq j$, $\text{var}(\mathbf{Y}_i(\mathbf{x})) = \mathbf{C}_k$ and $\mathbf{m} = E[\mathbf{Z}(\mathbf{x})]$ constant. An equivalent expression to Eq. (5.2) also exists if the mean cannot be assumed constant.

The *linear model of coregionalisation* (*LMC*, Wackernagel (2003)) is a linear combination of univariate permissible variogram functions g_k:

$$\boldsymbol{\Gamma}(\mathbf{h}) = \sum_{k=0}^{K} \mathbf{C}_k g_k(\mathbf{h}). \tag{5.5}$$

The coefficients $\mathbf{C}_k$ are matrices that are often constrained to be positive semi-definite, as the easiest way to ensure positive definiteness of the variance–covariance matrix of any linear combination of the variables. In the case when the variables are

second order stationary, then $\mathbf{C}(\mathbf{h}) = \mathbf{C}(\mathbf{0}) - \boldsymbol{\Gamma}(\mathbf{h})$ and so the covariance function may be used in place of the variogram to characterise the spatial continuity.

For a compositional random field the dependence structure can be specified in terms of a variogram (or covariance function) of the log-ratio transformed field (Sect. 4.3.3), or with a variation-variogram $\mathbf{T}(\mathbf{h})$, as explained in Sect. 4.3.4. In terms of variation-variograms, the LMC can be expressed as

$$\mathbf{T}(\mathbf{h}) = \sum_{k=0}^{K} \mathbf{B}_k g_k(\mathbf{h}),$$

where the matrices $\mathbf{B}_k$ are symmetric, zero-diagonal matrices with one positive eigenvalue (and the rest, negative). These are the conditions that make each $\mathbf{B}_k$ be a valid variation matrix (Eq. 3.3).

For a model to be valid, it is necessary that the semivariogram function is a conditionally negative definite function, and as this condition is hard to verify in practice, the LMC is the most commonly used model for cokriging and cosimulation. Its popularity is a consequence of the relative ease of modelling and verification of the sufficient admissibility of the model. However, the LMC has several limitations as it implies symmetrical cross-covariances for the variables under study. This property may not always be satisfied, nor do secondary and primary variables necessarily have the same support. To overcome this limitation Marcotte (2012) introduced a generalized LMC where the observed variables are modelled as linear combinations of underlying independent variables and of variables representing deterministic functions of some of these underlying variables. In the case of a regionalised composition the question of differences in support size does not arise; however, a lack of symmetry in cross-covariances is an unexplored issue.

5.2.2 *Log-Ratio Invariance of the LMC*

The LMC, being symmetric, can be expressed in any of these specifications

$$\mathbf{C}^{\psi}(h|\boldsymbol{\theta}) = \sum_{k=0}^{K} \mathbf{C}_k^{\psi} \cdot \rho_k(h|\theta_k), \qquad \mathbf{C}^{c}(h|\boldsymbol{\theta}) = \sum_{k=0}^{K} \mathbf{C}_k^{c} \cdot \rho_k(h|\theta_k), \tag{5.6}$$

$$\boldsymbol{\Gamma}^{\psi}(h|\boldsymbol{\theta}) = \sum_{k=0}^{K} \mathbf{C}_k^{\psi} \cdot g_k(h|\theta_k), \qquad \boldsymbol{\Gamma}^{c}(h|\boldsymbol{\theta}) = \sum_{k=0}^{K} \mathbf{C}_k^{c} \cdot g_k(h|\theta_k),$$

$$\mathbf{T}(h|\boldsymbol{\theta}) = \sum_{k=0}^{K} \mathbf{B}_k \cdot g_k(h|\theta_k),$$

where K denotes the number of structures, each with its correlogram $\rho_k(\cdot|\theta_k)$ or its unitary variogram $g_k(\cdot|\theta)$, and $\boldsymbol{\theta}$ denotes the vector of model parameters containing

all individual ranges θ_k and sill matrices. In the case of second order stationary functions, unitary variograms and correlograms are related through $g_k(\cdot) = 1 - \rho_k(\cdot)$. These sills can be specified either as $\mathbf{B}_k$ or $\mathbf{C}_k$, as they are equivalent because of the relations

$$\mathbf{C}_k^{\psi} = \boldsymbol{\Psi} \cdot \mathbf{C}_k^{c} \cdot \boldsymbol{\Psi}^t, \qquad \mathbf{C}_k^{c} = \boldsymbol{\Phi}^t \cdot \mathbf{C}_k^{\psi} \cdot \boldsymbol{\Phi}, \tag{5.7}$$

$$\mathbf{C}_k^{\psi} = -\frac{1}{2}\boldsymbol{\Psi} \cdot \mathbf{B}_k \cdot \boldsymbol{\Psi}^t, \qquad \mathbf{C}_k^{c} = -\frac{1}{2}\boldsymbol{H} \cdot \mathbf{B}_k \cdot \boldsymbol{H} \tag{5.8}$$

$$\mathbf{B}_k = \mathbf{J}\text{diag}(\mathbf{C}_k^{c}) + \text{diag}(\mathbf{C}_k^{c})\mathbf{J} - 2\mathbf{C}_k^{c}, \tag{5.9}$$

with $\mathbf{1}_{(D,D)} = \mathbf{1}_D\mathbf{1}_D^t$ a $D \times D$-matrix of ones. These relations mirror those given in Sect. 4.3.5 regarding the various representations of the theoretical variographic structure (in log-ratios, in variation-variograms, in covariance). The package "`compositions`" implements several of them: `clrvar2ilr` and `ilrvar2clr` allow moving between $\mathbf{C}_k^{\psi}$ and $\mathbf{C}_k^{c}$ (only for the ilr transformation!) through Eq. (5.7); `variation2clrvar` can be used to obtain $\mathbf{C}_k^{c}$ from $\mathbf{B}_k$ (Eq. 5.8); and `clrvar2variation` implements Eq. (5.9) to obtain $\mathbf{B}_k$ from $\mathbf{C}_k^{c}$.

A model must be positive definite if specified in terms of $\mathbf{C}^{\psi}(\cdot)$ and positive semi-definite in terms of $\mathbf{C}^{c}$ because the clr-transformed covariances always have at least one zero eigenvalue. Equivalently, a model specification in $\boldsymbol{\Gamma}^{\psi}(\cdot)$ must be conditionally negative definite, and in terms of $\boldsymbol{\Gamma}^{c}(\cdot)$ conditionally negative semi-definite. Thus, a model specified in $\mathbf{T}(\cdot)$ must be conditionally positive semi-definite.

For the LMC, these conditions are satisfied if:

1. Each correlogram $\rho_k(h|\theta_k)$ or unitary variogram is an admissible function $g_k(h|\theta_k)$, as e.g. those found in Table 5.1.
2. Each matrix $\mathbf{C}_k$ is a valid covariance matrix (a symmetric positive semi-definite matrix Wackernagel, 2003) or $\mathbf{B}_k$ is a variation matrix (a matrix with zero-diagonal elements being conditionally negative definite, i.e. having all but one negative eigenvalue).

These are sufficient, but not necessary conditions.

5.2.3 *Practical Modelling Procedure*

In a classical context fitting an LMC is aided by the following process (Goovaerts, 1997) and this also works for the log-ratio variograms:

1. Direct variograms are modelled first, but with the smallest number of basic structures that capture the essential features of all of them. There must be at least one common structure since the cross-variogram models may contain only

those structures that are present in the models of the direct variograms of all the variables involved.
2. The cross-variogram models are fitted using the same structures as the direct variograms. Note that it is not required to use all of the ones used for the direct variograms.
3. Consider anisotropy for a structure only if it is clearly visible in all directional variograms.
4. Assess the fit visually, if a compromise in the choice of structure is needed, bearing in mind that the direct variograms need to be given priority over cross variograms.

Most software will optimise the fit based on some least squares criterion, but it is critical to assess the quality of the fit before proceeding to use the LMC in any estimation or simulation procedure.

In a compositional context, one can follow the classical procedure if the use of a particular log-ratio transformation is considered as reasonable. If several log-ratio transformations are equally valid, care must be taken when fitting models to their empirical variograms, as it is necessary to force the fitted model to satisfy the conditions of Eqs. (5.7–5.9) too. In this case, the fitted models are said to be *mutually compatible*. In practice, ensuring compatibility during the modelling process may not be easy, as each variogram or covariance function derived from a particular log-ratio transformation may focus on some specific subcompositional features, while others might appear to mask the variability of that particular feature. It is thus possible that classical automatic fitting processes do not produce compatible models.

If a more general approach is preferred, then the fit of the LMC should be obtained in terms of the variation-variogram. Individual variation-variograms can be estimated with any available procedure for estimation of direct variograms. Fitting must nevertheless be done jointly, to ensure that the validity conditions stated in last section are satisfied.

Fitting models for the variation-variograms have certain advantages. Most importantly, it allows the use of a logarithmic goodness-of-fit criterion that minimises the function (Tolosana-Delgado et al., 2011),

$$\text{gof}(\theta) = \sum_{i=1}^{D} \sum_{j=1}^{i-1} \sum_{n=1}^{N_{ij}} (\ln \hat{t}_{ij}(\mathbf{h}_n) - \ln t_{ij}(\mathbf{h}_n|\theta))^2, \tag{5.10}$$

where $\hat{t}_{ij}(\mathbf{h}_n)$ is the empirical (i, j)-variation-variogram calculated for a lag distance class $\mathbf{h}_n$, N_{ij} is the number of lags used for that calculation, and $t_{ij}(\mathbf{h}_n|\theta)$ is the model evaluated at lag h_n. The choice of Eq. (5.10) as a fit criterion focuses the fitting procedure on the smaller values of t_{ij}, that is, around the origin and at short ranges. Two arguments support this choice. First, interpolation results are particularly sensitive to this part of the variogram, and they do not really depend on the sill: spending effort on fitting the sill will not translate to more reliable interpolations (Chilès & Delfiner, 2012). Second, as sills are variances, their natural

spread ranges in orders of magnitudes, or in other words, they should be compared in a relative scale, thus in logarithms: this view is consistent with the multiplicative confidence intervals given for the sill of a variogram (Cressie, 1991).

As a second advantage, fitting variation-variograms enable working with data sets with many missing values, partially observed subcompositions and similar irregularities. As each component $t_{ij}(\mathbf{h}_n)$ requires only that variables i and j are available, one uses pairwise elimination to calculate $\hat{t}_{ij}(\mathbf{h}_n)$, thus using a maximum number of observations for each lag and each pair of variables. In contrast, estimating variograms with ilr- or clr-transformed data requires eliminating much more data: for instance, the clr vector is not available if even one single component is missing (i.e. one would need complete row-wise elimination to deal with missing values); and though an ilr can be chosen to maximise the number of computable log-ratio scores, this only works if there are only a few components with missing values (Tolosana-Delgado et al., 2008). One could argue that the set of estimates $\{\hat{t}_{ij}(\mathbf{h})\}$ obtained in this way does not necessarily define a set of valid variation matrices $\{\hat{\mathbf{T}}(\mathbf{h}_n)\}$, being computed from different subsets of the data. However, in a spatial structural analysis this is of no consequence, because the set $\{\hat{t}_{ij}(\mathbf{h}_n)\}$ is only used to guide the fit of a parametric model $\mathbf{T}(\mathbf{h}|\theta)$ to the data, which will in any case be obtained with an algorithm embedding the validity constraints on $\mathbf{T}(\mathbf{h}|\theta)$ explicitly.

For these reasons, a structural analysis of regionalised compositional data in terms of variation-variograms is desirable. One could use standard estimation algorithms to obtain each individual pairwise log-ratio variogram. Then, a procedure would be required for fitting an LMC using Eq. (5.10), subject to the condition that each matrix $\mathbf{B}_k$ is a valid variation matrix (zero diagonal, symmetric, one positive eigenvalue and the rest negative eigenvalues). Finally, the fitted model should be compared to the empirical structural functions, either in terms of the variation-variograms or recasting variogram estimates and model to clr-/ilr variograms or covariances, using Eq. (5.8).

5.3 Model Functions and Model Fitting

5.3.1 *Packages "gmGeostats", "compositions" and "gstat"*

As explained in Chap. 4, packages "gstat" and "gmGeostats" require the establishment of a data container (with functions `gstat`, respectively, `make.gmCompositionalGaussianSpatialModel`), which can then be used to feed the spatial functions of the package. In particular, function `variogram` could be used to compute empirical variograms, e.g.

```
> gg = make.gmCompositionalGaussianSpatialModel(
+   acomp(compo), coords, V="alr", formula = ~1)
> vg = variogram(gg, cutoff=50, width=3.5)
```

Alternatively we have also seen ways for producing variation-variograms with the function `logratioVariogram` from the package "`compositions`" and that these can be transformed into log-ratio variograms with the command `as.gstatVariogram`, e.g.

```
> lrvg = logratioVariogram(acomp(compo), coords, maxdist=50)
> vg = as.gstatVariogram(lrvg, V="alr")
```

Following these calculations of experimental variograms, the model structure needs to be specified. In "`gmGeostats`" this is done for compositional data via the command `LMCAnisCompo`; in "`gstat`", using command `vgm`. For example, a model comprised a nugget and a single exponential structure of unknown range and sill can be quickly specified in "`gstat`" by

```
> md0=vgm("Exp")
```

but the same function also allows specification of nested structures, or initial guesses of the ranges and partial sills of the structures

```
> md0=vgm("Exp", range=50, psill=1,
+         add.to=vgm("Sph", range=20, nugget=1, psill=1))
```

Function `LMCAnisCompo` can be used similarly to define variograms with different degrees of specificity, albeit always including the original data:

```
> LMCAnisCompo(acomp(wind.compo), models="sph")
> LMCAnisCompo(acomp(wind.compo),
+         models=c("nugget","sph"), ranges=c(0, 50))
```

These provide two different ways for creating such objects, only specifying the shape (first row), or setting shape and ranges (second row).

The models of Table 5.1 are available, with the names given in Table 5.2. Many more are available; calls to `vgm()` or `show.vgms()` produce, respectively, a list of all names and a plot of each variogram shape in "`gstat`". Once the model is specified, it can be fit to the empirical variograms. This is done with function `fit_lmc`

```
> gg=fit_lmc(v=vg, g=gg, model=md0)
```

Note that this function can take all three objects that we just created, in which case it produces a copy of the root "`gstat`" object, with the fitted variogram model

Table 5.2 Names of the most important variogram models found in the packages "`gstat`" and "`compositions`". Package `gmGeostats` understands both specifications

Model	"`gstat`"	"`compositions`"
Nugget	(Implicit)	nugget
Spherical	"`Sph`"	sph
Exponential	"`Exp`"	exp
Gaussian	"`Gau`"	gauss
Generalised linear	"`Pow`"	pow
Stable	"`Exclass`"	(-)
Cardinal sine	"`Wav`"	cardsin

appended. Or it can be called only with empirical and theoretical variogram

```
> md=fit_lmc(v=vg, model=md0)
```

in which case only the fitted variogram is returned. This second usage is only available if "gmGeostats" is loaded.

Package "compositions" also has a series of functions for geostatistical analysis. For variography, the main functions are `logratioVariogram` (explained in Sect. 4.3.4), `CompLinModCoReg`, responsible for establishing the structures of the LMC, and again a method for `fit_lmc`, implementing the fit with the criterion given in Eq. (5.10). The workflow to obtain a fitted LMC with these functions starts with the calculation and plot of the variogram for a maximum distance and a number of lags,

```
> pwlr.vg = logratioVariogram(acomp(compo), coords, maxdist =50,
+                             nbins =20 )
> plot(lrvg)
```

The following step is the specification of the LMC

```
> clr.md = CompLinModCoReg(formula=~nugget()+sph(10),
+                                          acomp(compo))
```

Here the `formula` argument is a formula without left hand term that gives a sum of the models to be considered in the LMC. In the example above, we would consider a nugget and a spherical structure of range 10 m. All models of Table 5.1 are implemented, with the symbolic names given in Table 5.2. Once the model is defined, it can be fitted to the experimental variogram

```
> clr.lmc = fit_lmc(pwlr.vg, clr.md)
```

The function `fit.lmc` provides a method for variation-variograms, returning a function of class "CompLinModCoReg", with an extra attribute `fit.quality` reporting the non-linear minimisation used for fitting the LMC. The main part of the output is a functional clr-variogram representation of the fitted LMC (Eq. 5.6). This needs to be converted with Eq. (5.9) into a variation-variogram specification before plotting. The package provides a function `vgram2lrvgram` to do that conversion

```
> pwlr.lmc = vgram2lrvgram(clr.lmc)
> variogramModelPlot(pwlr.vg, pwlr.lmc)
```

The geostatistical functions of package "compositions" are not able to deal with anisotropy. For this reason, package "gmGeostats" incorporates extensions of the functions mentioned before. Function `logratioVariogram` was already used in Sect. 4.3.6

```
> pwlr.vgA = logratioVariogram(acomp(compo), coords,
+                                        azimuth.tol = 22.5)
```

As mentioned in the preceding section, the LMC can be specified with the function `LMCAnisCompo`, to be fitted to "`logratioVariogramAnisotropy`" objects by means of yet another method of the function `fit_lmc`

```
> clr.lmc = fit_lmc(pwlr.vgA, model=clr.lmc)
```

From a practical point of view, thus, the variographic analysis with anisotropy can be indistinctively done with any of the specifications mentioned before (`LMCAnisCompo`, `vgm`, `CompLinModCoReg` for the model vs. `variogram` or `logratioVariogram` for empirical variograms). The two common functions (`variogramModelPlot` and `fit_lmc`) can use any combination of one empirical and one model variogram, to plot both or to fit the model; and the various conversion tools provided can help in moving between different representations: `as.variogramModelList` produces a "`gstat`" variogram model specification, while `as.LMCAnisCompo` produces a variation-variogram LMC specification. The usage of these functions will be illustrated in Sect. 5.5.

5.4 Factorial Representations

Inference of an LMC can be quite cumbersome. At times it is easier to circumvent this step and instead work with PCA or MAF factors, in particular if their spatial decorrelation measures (Sect. 4.5.1) are good enough. The idea here is to fit models to each of the factors and then derive an LMC through matrix arithmetic.

5.4.1 PCA

To compute factors based on the PCA, the spectral decomposition of the (positive definite) variance–covariance matrix $\mathbf{S}$ of the variables transformed to log-ratios is computed using Eq. (3.5), and the log-ratio transformed variables are transformed according to

$$\hat{\boldsymbol{\zeta}}^{(pc)} = \boldsymbol{\Lambda}^{-1/2}\mathbf{Q}\boldsymbol{\zeta}^{(lr)}. \tag{5.11}$$

If the LMC for the raw data is given as

$$\boldsymbol{\Gamma}(\mathbf{h}) = \sum_{k=0}^{K} \mathbf{C}_k \gamma_k(\mathbf{h}) \tag{5.12}$$

with $\sum_{k=0}^{K} \mathbf{C}_k = \mathbf{S}$, then $\sum_{k=0}^{K} \boldsymbol{\Lambda}^{-1/2}\mathbf{Q}^t\mathbf{C}_k\mathbf{Q}\boldsymbol{\Lambda}^{-1/2} = \mathbf{I}$, i.e. the factors are uncorrelated at lag zero. But there is no guarantee that cross variograms for lag vectors $\mathbf{h} \neq \mathbf{0}$ are equal to 0. This is only true in the case of *intrinsic correlation* (Wackernagel, 2003), that is, when the coregionalisation matrices $\mathbf{C}_k$ are multiples of $\mathbf{S}$, in which case all factors exhibit the same spatial structure. In general we would like to achieve not just a decorrelation at 0 separation distance, but also for non-zero lag vectors.

5.4.2 MAF

Given an experimental MAF decomposition (4.4) of the log-ratio data into factors, one can fit an (univariate) LMR for each of the factors. The resulting variograms can then be combined to a joint model. If p structures are needed for describing the spatial variability of all factors, then the combined model for the factors is given by

$$\boldsymbol{\Gamma}_{MAF}(\mathbf{h}) = \sum_{i=1}^{p} \mathbf{D}_i g_i(\mathbf{h}),$$

where the matrices $\mathbf{D}_k$ are diagonal matrices by construction. They do not necessarily have full rank. If for the kth variogram structure factor i is absent, then $d_{ii}^k = 0$. The LMC for the original data can then be constructed as

$$\boldsymbol{\Gamma}(\mathbf{h}) = \mathbf{W}^t \boldsymbol{\Gamma}_{MAF}(\mathbf{h})\mathbf{W} = \sum_{i=1}^{p} \mathbf{W}^t \mathbf{D}_i \mathbf{W} g_i(\mathbf{h}), \tag{5.13}$$

where $\mathbf{W} = \mathbf{A}^{-1}$ is the inverse of the loadings matrix $\mathbf{A}$ of the MAF transformation. In the case of compositional variables, care needs to be taken because $\mathbf{A}$ is a non-square matrix; then $\mathbf{W} = \mathbf{A}^-$ is the generalised inverse of $\mathbf{A}$, i.e. a matrix such that $\mathbf{W} = \mathbf{WAW}$ and $\mathbf{A} = \mathbf{AWA}$. The function `Maf` takes care of this eventuality and provides an appropriate generalised inverse of the MAF within the element `invLoadings`

```
> A = maf.wind$loadings
> W = maf.wind$invLoadings
> A %*% W %*% A - A
```

```
            [,1]       [,2]       [,3]       [,4]       [,5]
Fe     2.22e-16 -8.88e-16  3.55e-15 -8.88e-16 -3.55e-15
Al2O3  3.33e-16 -8.88e-16  2.66e-15 -9.44e-16 -4.88e-15
SiO2  -6.66e-16  9.30e-16 -4.66e-15  1.44e-15  4.22e-15
Mn     3.33e-16 -3.33e-16 -7.22e-16 -1.11e-16 -8.33e-17
P      2.22e-16 -4.44e-16  6.44e-15 -1.11e-15 -5.22e-15
R     -4.44e-16  1.33e-15 -1.07e-14  2.66e-15  1.24e-14
```

These calculations require the MAF decomposition object computed on page 67.

From a practical point of view, in the modelling step care should be taken to maximise the number of joint variogram structures in terms of range and shape to ensure a parsimonious model is obtained. In particular, if the underlying LMC can be modelled with just two structures, then the MAF perfectly captures the whole variability (Rondon, 2012).

Given the analogy between $\mathbf{A}$, $\mathbf{Q}$ and $\boldsymbol{\Psi}$ highlighted in Sect. 4.4, a MAF representation of a compositional random function is obtained by simply giving the loadings

matrix **A** of the MAF analysis to the representation function `compo2gstatLR`

```
> maf = Maf(compo, lrvg, i)
> maf.gg = make.gmCompositionalGaussianSpatialModel(
+   compo, coords, V=maf$loadings)
```

The same idea can be applied for principal component analysis, i.e.

```
> pca = princomp(compo)
> pca.gg = make.gmCompositionalGaussianSpatialModel(
+   compo, coords, V=pca$loadings)
```

5.4.3 Rank-One Structures

An interesting case of factorisation of the variographic structure occurs if the linear model of coregionalisation can be expressed as

$$\boldsymbol{\Gamma}(\mathbf{h}) = \sum_{k=0}^{K} \mathbf{w}_k \mathbf{w}_k^t g_k(\mathbf{h}), \tag{5.14}$$

where $\mathbf{w}_k$ are individual vectors, each associated with a different structure $g_k(\cdot)$. In the case of real-valued P-variate random function, each $\mathbf{w}_k \in \mathbb{R}^P$, while in the case of a $(P+1)$-part compositional random function they will represent clr-transformed compositions, i.e. $\mathbf{w}_k \in \mathbb{R}^{P+1}$ and $\sum_{j=1}^{P+1} w_{jk} = 0$ for each k.

The interest of this representation lies in the case where $K \leq P$. In that case, the matrix $\mathbf{A} = \mathbf{W}^-$, the generalised inverse of $\mathbf{W}$, has the interesting property of diagonalising the covariance structure, because by definition $\mathbf{AW} = \mathbf{I}_{(K,K)}$, and $\mathbf{Aw}_k = \mathbf{i}_k$ the k-th column vector of the identity matrix $\mathbf{I}_{(K,K)}$

$$\mathbf{A}\boldsymbol{\Gamma}(\mathbf{h})\mathbf{A}^t = \sum_{k=0}^{K} \mathbf{Aw}_k \mathbf{w}_k^t \mathbf{A}^t g_k(\mathbf{h}) = \sum_{k=0}^{K} \mathbf{i}_k \mathbf{i}_k^t g_k(\mathbf{h}). \tag{5.15}$$

Note that $\mathbf{i}_k \mathbf{i}_k^t$ is a matrix with a 1 on the k-th diagonal element and 0 in all other entries. This produces an extreme decorrelation between the factors represented by the matrix **A**.

Rank-one representations of the LMC can be obtained with the geostatistical functions of "`compositions`", while specifying the formula in `CompLinModCoReg`: if one desires a certain variogram to be associated with a rank-one covariance, that structure needs to be multiplied by the symbol `R1`. Thus one could consider a model such as

```
> pwlr.md = CompLinModCoReg(formula=~nugget()+R1*exp(30),
+                           acomp(compo))
```

meaning that the nugget can be a regular full rank structure, but the $30m$-range exponential should be given a rank-one structure. The model can be fitted by means of the command `fit_lmc` as explained at the end of Sect. 5.3.1.

5.5 Example: Variogram Models for the Windarling Data

In this section an LMC for the Windarling composition will be fitted, following the calculations in pages 57–59, and the scheme of Sect. 5.3.1. The first step is the representation of the compositional random function in log-ratios, for instance in alr, followed by the calculation of the empirical variogram

```
> wind.alr.vg = variogram(wind.alr.gg, cutoff=50, width=3.5)
```

We then define a variogram model template with function `vgm` and fit it with `fit_lmc`

```
> md0=vgm("Exp")
> wind.alr.gg = fit_lmc(v=wind.alr.vg, g=wind.alr.gg, model=md0)
```

In the case of the Windarling data we obtain the model shown in Fig. 5.2.

```
> plot(wind.alr.vg, wind.alr.gg$model)
```

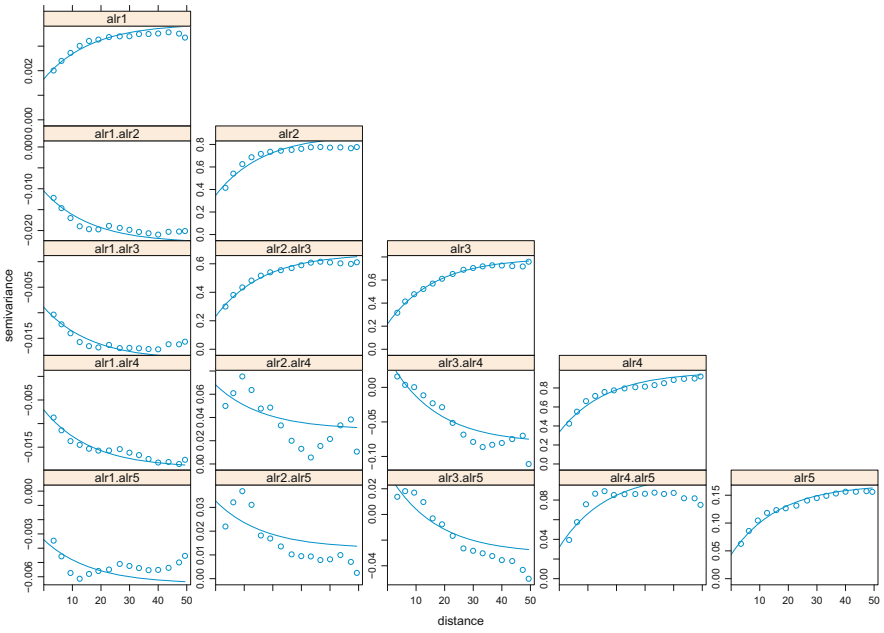

Fig. 5.2 Autofitted isotropic LMC for Windarling composition comprised of a nugget and an exponential structure

Table 5.3 Sill matrices of nugget (first 5 rows) and exponential structure (last 5 rows) of the isotropic LMC for Windarling, also shown in Fig. 5.2

	alr_Fe	alr_Al2O3	alr_SiO2	alr_Mn	alr_P
alr_Fe	0.0017	−0.0105	−0.0089	−0.0071	−0.0034
alr_Al2O3	−0.0105	0.3478	0.2304	0.0681	0.0327
alr_SiO2	−0.0089	0.2304	0.2187	0.0442	0.0330
alr_Mn	−0.0071	0.0681	0.0442	0.3349	0.0320
alr_P	−0.0034	0.0327	0.0330	0.0320	0.0432
alr_Fe	0.0023	−0.0125	−0.0103	−0.0123	−0.0031
alr_Al2O3	−0.0125	0.5398	0.4412	−0.0387	−0.0200
alr_SiO2	−0.0103	0.4412	0.5744	−0.1254	−0.0634
alr_Mn	−0.0123	−0.0387	−0.1254	0.6349	0.0751
alr_P	−0.0031	−0.0200	−0.0634	0.0751	0.1261

The sill parameters for this fitted LMC are tabulated in Table 5.3 and the range of the exponential structure is roughly 16.5 m.

```
> wind.alr.gg$model$alr1$range

[1]  0.0 16.5
```

The model is not yet entirely satisfactory, as a short scale structure present in the cross variograms of alr2, alr4 and alr5 is smoothed out. This issue can be overcome by specifying ranges and experimenting with the types of spatial structures. For example the model in Fig. 5.3 provides a better fit of the short scale structure evident in the cross variograms for alr2, alr4 and alr5.

```
> md1 = vgm(psill=1, model="Sph", range=15, nugget=1)
> md1 = vgm(psill=1, model="Sph", range=45, add.to=md1 )
> wind.alr.gg = fit_lmc(v=wind.alr.vg, g=wind.alr.gg, model=md1,
+                       correct.diagonal = 1.001)
> plot(wind.alr.vg, wind.alr.gg$model)
```

For this LMC the sill parameters are tabulated in Table 5.4 and the ranges are, respectively,

```
> wind.alr.gg$model$alr1$range

[1]  0 15 45
```

i.e. as specified in the beginning. This highlights that a standard use of `fit_lmc` does not adjust the ranges but will fit the sills. Sometimes, it is more convenient to look at these models as truly multivariate, and not as a collection of individual models (as they are actually stored in "`gstat`"). To do this, we need to convert them into the variogram model system of "`gmGeostats`", with function `as.LMCAnisCompo`

```
> (LMC_A = as.LMCAnisCompo(wind.alr.gg, V="alr"))

      [,1]       [,2]       [,3]
model "nugget"   "sph"      "sph"
```

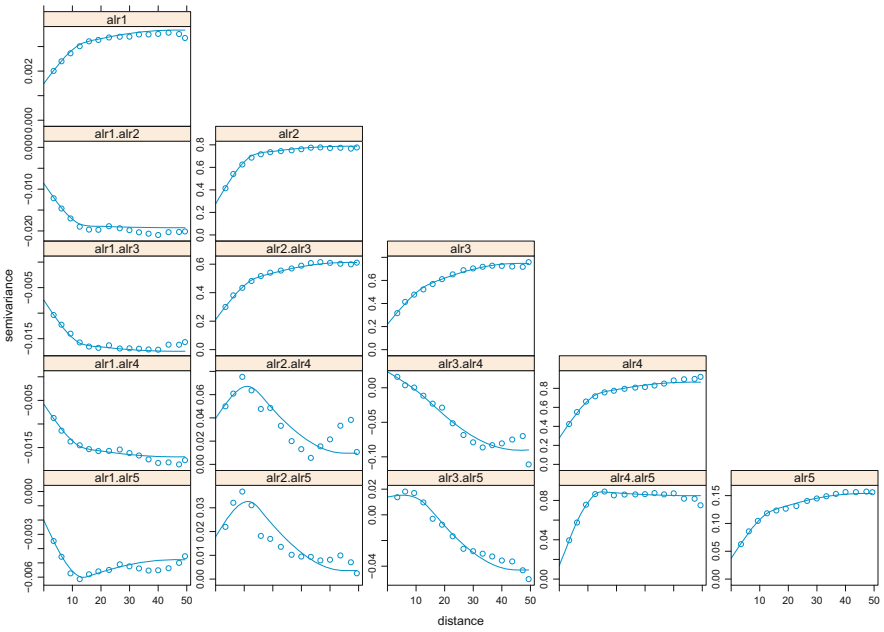

Fig. 5.3 Isotropic LMC for Windarling composition comprised of a nugget and two spherical structures

Table 5.4 Sill matrices of nugget (first 5 rows), short-range (rows 6 to 10) and long-range (last five rows) spherical variogram structures for Windarling, also represented in Fig. 5.3.

	alr_Fe	alr_Al2O3	alr_SiO2	alr_Mn	alr_P
alr_Fe	0.0015	−0.0086	−0.0074	−0.0058	−0.0020
alr_Al2O3	−0.0086	0.2762	0.2112	0.0395	0.0178
alr_SiO2	−0.0074	0.2112	0.2203	0.0232	0.0139
alr_Mn	−0.0058	0.0395	0.0232	0.2775	0.0138
alr_P	−0.0020	0.0178	0.0139	0.0138	0.0371
alr_Fe	0.0013	−0.0097	−0.0078	−0.0079	−0.0051
alr_Al2O3	−0.0097	0.3924	0.2048	0.0704	0.0366
alr_SiO2	−0.0078	0.2048	0.1902	0.0191	0.0343
alr_Mn	−0.0079	0.0704	0.0191	0.3826	0.0786
alr_P	−0.0051	0.0366	0.0343	0.0786	0.0603
alr_Fe	0.0009	−0.0009	−0.0022	−0.0032	0.0023
alr_Al2O3	−0.0009	0.1187	0.1957	−0.1003	−0.0509
alr_SiO2	−0.0022	0.1957	0.3363	−0.1323	−0.0911
alr_Mn	−0.0032	−0.1003	−0.1323	0.2045	−0.0075
alr_P	0.0023	−0.0509	−0.0911	−0.0075	0.0562

```
range 0          15         45
A     Numeric,9  Numeric,9  Numeric,9
sill  Numeric,36 Numeric,36 Numeric,36
attr(,"class")
[1] "LMCAnisCompo"
```

The resulting object appears to be a matrix of complex elements, among others including sills, model shapes and anisotropic ranges. This may be useful, for instance, to quickly check the dimensionality of the various structures, e.g. with an eigenvalue decomposition. In the Windarling case, the eigenvalues of the three sill matrices indicate that all structures are of rank 5, i.e. the number of alr components

```
> variationSills = LMC_A["sill",]
> sapply(variationSills, dim)

     [,1] [,2] [,3]
[1,]    6    6    6
[2,]    6    6    6

> sapply(variationSills, function(x)
+   eigen(variation2clrvar(x))$values)

          [,1]     [,2]      [,3]
[1,] 3.22e-01 3.79e-01  5.17e-01
[2,] 1.95e-01 2.48e-01  1.34e-01
[3,] 3.52e-02 5.93e-02  8.69e-03
[4,] 2.58e-02 3.05e-02  5.72e-04
[5,] 4.94e-04 3.08e-04  7.12e-05
[6,] 1.24e-17 2.37e-17 -2.14e-17
```

To get that result the "gstat" LMC description was first recast into a variation-variogram LMC object. This is necessary because then the sills are easily extracted as a list of as many elements as the number of structures, each containing a variation matrix of size 6×6. We have learnt this using the command `sapply` onto the variation sills list together with the function `dim`: this yields the result of applying the function to each element of the list. Finally, the last line uses the same trick to apply an ad hoc function to the variation sills list: this function extracts the eigenvalues of the clr-variance matrix linked to each variation sill, obtained with Eq. (5.8). Note that as discussed in Sect. 5.2.2, clr-variances should have a zero eigenvalue that is shown in the sixth row of these results.

Since the experimental variograms computed in Chap. 4 exhibited anisotropy the fitting of an anisotropic variogram is also required. For the Windarling data the direction of greatest continuity is roughly EW, which corresponds to an azimuth angle of $N90^o$. The "gstat" specifications for the experimental variograms that are now required are the directions to be computed, starting with that of greatest continuity and the angular tolerance. These are specified via the parameters `alpha` and `tol.hor` in the `variogram` command.

```
> wind.alr.vgAnis = variogram(wind.alr.gg, cutoff=50, width=5,
+                             alpha=c(90,180), tol.hor=45)
> plot(wind.alr.vgAnis,group.id=FALSE,col=c("red","black"))
```

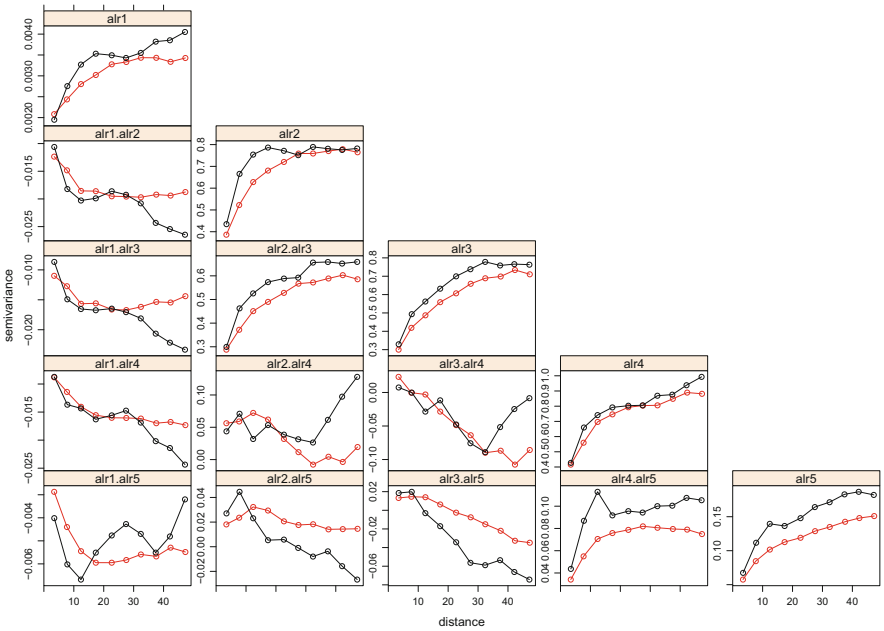

Fig. 5.4 Experimental directional direct and cross variograms in directions N90 (red) and N180 (black) for Windarling composition

Inspection of the experimental semivariograms in Fig. 5.4 suggests an isotropic short-range structure and another structure with ranges of approximately 60 m in direction N90 and 30 m in direction N180. Alternatively, an intermediate exponential structure with long range 30 m (E-W) and 12 m (N-S). These ranges are declared by setting the range in the direction of greatest continuity and the anisotropy ratio. Sills will be fitted automatically.

```
> mdA =
+   vgm(psill=1, model="Exp", range=30, nugget=1, anis=c(90,.4))
> wind.alr.ggAnis = fit_lmc(v=wind.alr.vgAnis, g=wind.alr.gg,
+                           model=mdA, correct.diagonal = 1.001)
> variogramModelPlot(wind.alr.vgAnis, wind.alr.ggAnis)
```

The model, shown in Fig. 5.5, is quite unsatisfactory. For example the cross-variogram model for the experimental cross variogram of alr3 and alr1 is a pure nugget model and the shapes of the experimental cross variograms for alr1 and alr5, and for alr2 and alr5 are not adequately reproduced. We will therefore adopt a different strategy to the fitting of the LMC. We will use the data driven MAF derived in Chap. 4. This is done by first fitting anisotropic variogram models to the directional experimental variograms of the MAF factors and then using the models derived to build a joint LMC.

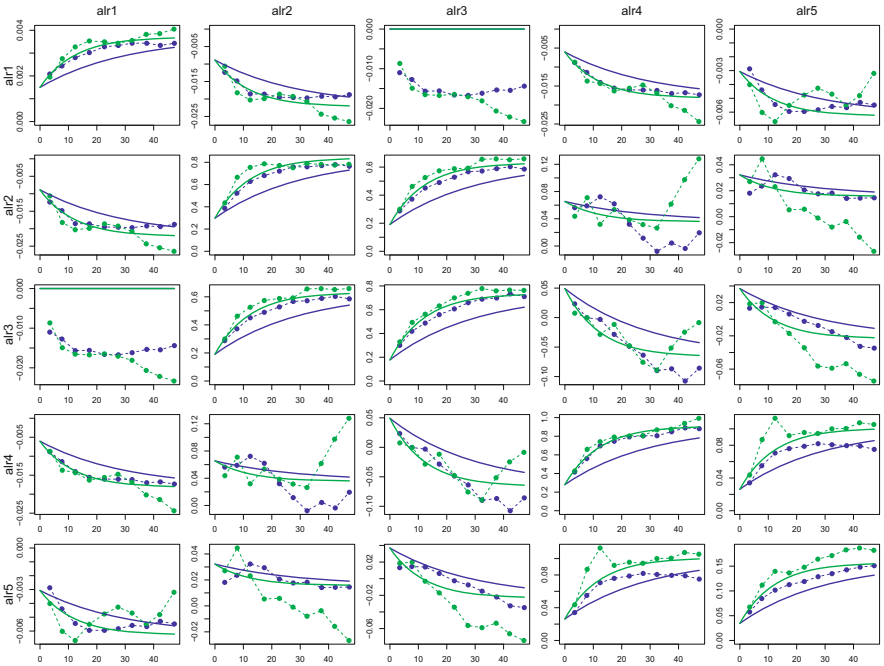

Fig. 5.5 Anisotropic LMC for Windarling composition comprised of a nugget and an exponential structure

The first step is the representation of the composition in MAF scores

```
> wind.maf.ggAnis =
+  make.gmCompositionalGaussianSpatialModel(data = wind.compo,
+                    coords = wind.coords, V = maf.wind$loadings,
+                    prefix="maf")
```

Then the scores of the first maf are used to create a container object, and directional experimental variograms are computed for the two directions known to be relevant in this data set

```
> wind.maf1.gg = gstat(id="maf1", formula=maf1~1,
+                      data=wind.maf.ggAnis)
> wind.maf1.vg = variogram(wind.maf1.gg, cutoff=50,
+                      width=3.5, alpha=c(90,180))
```

By means of `plot(wind.maf1.vg)` and some trial and error, a variogram model with a satisfactory fit is obtained. It consists of a nested structure of a nugget and two spherical variograms of ranges 20 m and 70 m, respectively.

```
> wind.maf1.md = vgm(nugget=0.15, psill =0.2, range=10,
+            model = "Sph", anis=c(90, 1))
> wind.maf1.md = vgm(add.to=wind.maf1.md, psill =0.7, range=60,
+             model = "Sph", anis=c(90, 0.6))
> plot(wind.maf1.vg, wind.maf1.md)
```

The models for the remaining mafs are constructed similarly. Once satisfactory models for the spatial continuity of the MAF factors are available, they can be used to construct an LMC for the ilr data, which can then be recast to clr, alr or pwlr, as required. For the Windarling composition, in total 4 structures are needed to describe the spatial variation. These are a nugget, a short scale isotropic spherical structure of range 10 m, a spherical structure with geometric anisotropy of range 20 m in the direction N90 and anisotropy ratio 0.5 and a long scale spherical structure of range 70m in direction N90 and anisotropy factor 0.5. As a first step the sills need to be extracted from the models and stored in diagonal matrices. Then the model needs to be built from the sill matrices and the ranges. The results are a model for the pwlr data that can be recast to alr, ilr or clr as required.

```
> MAFm=matrix(0,ncol=5,nrow=6)
> rownames(MAFm)=c("Sp0","Sp10","Sp25","Sp40","Sp60","Sp70")
> aux=list(wind.maf1.gg, wind.maf2.gg, wind.maf3.gg,
+          wind.maf4.gg, wind.maf5.gg)
> for (i in 1:5) {
+   MAFm[paste("Sp",aux[[i]]$model[[1]]$range,sep=""),i]=
+     aux[[i]]$model[[1]]$psill}
> MAFanis=matrix(0,ncol=5,nrow=6)
> rownames(MAFanis)=c("Sp0","Sp10","Sp25","Sp40","Sp60", "Sp70")
> for (i in 1:5) {
+   MAFanis[paste("Sp",aux[[i]]$model[[1]]$range,sep=""),i]=
+     aux[[i]]$model[[1]]$anis1}
> MAFsills=apply(MAFm,1,diag)
> dim(MAFsills)=c(5,5,6)
> MAFranges=cbind(c(0,10,25,40, 60, 70),c(0,10,12.5,20,36,49))
> MAFazimuth=c(0,0,90,90,90,90)
```

The pwlr model is computed below and the model shown in Fig. 5.6. This represents a usage of `LMCAnisCompo` with almost all arguments possible .

```
> wind.pwlr.mdAnis=LMCAnisCompo(wind.compo,
+            models=c("nugget", "sph", "sph","sph","sph","sph"),
+            azimuths=MAFazimuth, ranges=MAFranges,
+            sillarray=MAFsills, V=maf.wind$loadings, tol=1e-12)
```

Note that the argument `V` expects the direct transformation matrix, i.e. the one that was used to pass from compositional data to whichever scores are used (log-ratio, PC scores or MAF scores); the function actually uses (and computes) the inverse of this matrix internally. Once `wind.pwlr.mdAnis`, the model in pwlr representation, is available, one can compute any other representation with `as.variogramModel`. The model fits for alr- and ilr-transformed data are shown in Figs. 5.7 and 5.8. In both cases the fit is superior to that achieved through a direct fitting of an LMC.

```
> plot(wind.pwlr.vgAnis, model=wind.pwlr.mdAnis,
+      azimuths=c("90N","180N"))

> wind.alr.mdAnis=as.variogramModel(wind.pwlr.mdAnis, V="alr")
> variogramModelPlot(wind.alr.vgAnis, wind.alr.mdAnis)
```

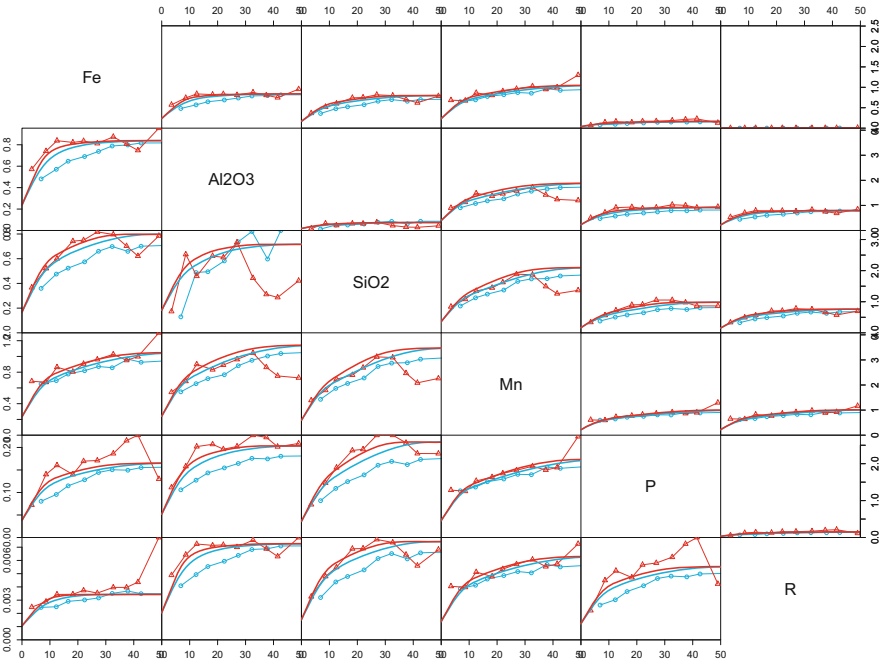

Fig. 5.6 LMC for Windarling composition derived from the MAF decomposition, in pwlr format

```
> wind.ilr.vgAnis = as.gstatVariogram(
+       wind.pwlr.vgAnis[,c("90N", "180N")],
+       V="ilr")
> wind.ilr.mdAnis=as.variogramModel(wind.pwlr.mdAnis, V="ilr")
> variogramModelPlot(wind.ilr.vgAnis, wind.ilr.mdAnis, xlim=c(0,50))
```

Problems

5.1 Variography for a Subcomposition of the Tellus Data

(a) For the Tellus subcomposition MgO, Al_2O_3, CaO, Fe_2O_3 and R, compute experimental omnidirectional direct and cross variograms in ilr coordinates.
(b) Fit an LMC to the experimental direct and cross variograms calculated in (a).

5.2 Verification of Adequacy of Fit in Different Log-Ratio Coordinate Representations

(a) Recast the ilr LMC in pwlr format and check that the fit is adequate for the experimental pwlr variograms.
(b) Recast the pwlr LMC in alr format and check that the fit is adequate for the experimental alr direct and cross variograms.

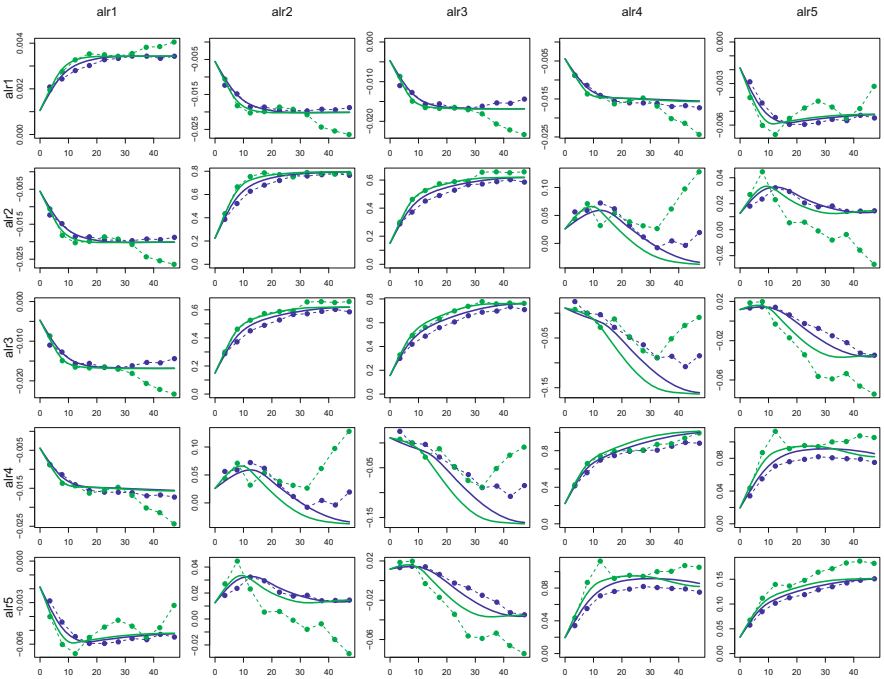

Fig. 5.7 LMC for Windarling composition in alr-coordinates derived from the MAF decomposition

5.3 Check of Model Fit for the Tellus Subcomposition on a Subset of the Data

Verify that the LMC in ilr coordinates calculated for the Tellus subcomposition MgO, Al_2O_3, CaO, Fe_2O_3 and R provides an adequate fit to direct and cross variograms of the subcomposition restricted to the subset of samples considered in Chap. 4.

5.4 LMC and Variogram Models for MAF Factors of the NGSA Major Elements

(a) Fit an isotropic LMC to the experimental direct and cross variograms of the NGSA major elements from `REGION=="EAST"` and `CODE=="Bc"` (bottom soil, coarse fraction) considered in Problem 4.2.
(b) Fit LMRs to variograms of the MAF factors of the composition in (a) and recast the resulting models as an isotropic LMC for the composition of major elements.
(c) Which of the models from (a) and (b) provides the better fit?

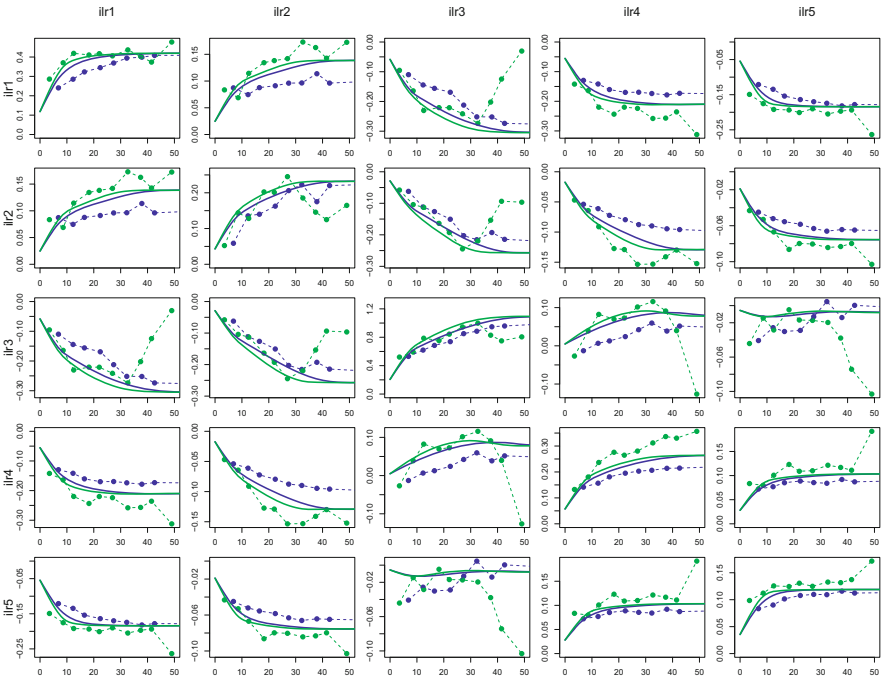

Fig. 5.8 LMC for Windarling composition in ilr-coordinates derived from the MAF decomposition

References

Chilès, J. P., & Delfiner, P. (2012). *Geostatistics - Modeling spatial uncertainty* (2nd ed., 699 pp.). Hoboken, NJ, USA: Wiley.

Cressie, N. (1991). *Statistics for spatial data* (900 pp.). New York, NY, USA: John Wiley and Sons.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation* (483 pp.). Applied Geostatistics Series. New York, NY, USA: Oxford University Press.

Marcotte, D. (2012). *Geostatistics Oslo 2012*, Chapter Revisiting the linear model of coregionalization (pp. 67–78). Quantitative Geology and Geostatistics. Springer.

Rondon, O. (2012). Minimum/maximum autocorrelation factors for joint simulation of attributes. *Mathematical Geosciences, 44*(4), 469–504.

Tolosana-Delgado, R., Egozcue, J. J., & Pawlowsky-Glahn, V. (2008). Cokriging of compositions: logratios and unbiasedness. In J. M. Ortiz, X. Emery (Eds.), *Geostatistics Chile 2008* (pp. 299–308). Gecamin Ltd., Santiago, Chile, 2 vols, 1188 p.

Tolosana-Delgado, R., Boogaart, K. G. v. d., & Pawlowsky-Glahn, V. (2011). Geostatistics for compositions. In V. Pawlowsky-Glahn, A. Buccianti (Eds.), *Compositional data analysis: Theory and applications* (pp. 73–86, 378 pp.). John Wiley & Sons.

Wackernagel, H. (2003). *Multivariate geostatistics: An introduction with applications* (293 pp.). Berlin: Springer.

Chapter 6
Geostatistical Estimation

Abstract In this chapter we consider the main geostatistical estimation methods adapted to regionalised compositions.

6.1 Cokriging

Once a valid LMC for the log-ratio variables is available, kriging estimates can be computed. These are computed in log-ratio coordinates and a suitable choice is alr or ilr. As in the case of classical multivariate geostatistics, estimates of the log-ratio variables can be made at unsampled locations using the covariance structure defined in (5.5). When the regionalised composition is second order stationary, the *cokriging estimate* $\boldsymbol{\zeta}^*_{SCK}(\mathbf{x}_0)$ at location $\mathbf{x}_0$ is given by

$$\boldsymbol{\zeta}^*_{SCK}(\mathbf{x}_0) = \bar{\boldsymbol{\zeta}} + \sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha(\mathbf{x}_0)[\boldsymbol{\zeta}(\mathbf{x}_\alpha) - \bar{\boldsymbol{\zeta}}], \tag{6.1}$$

where $\bar{\boldsymbol{\zeta}}$ denotes the global mean of the regionalised composition as defined earlier and $\mathbf{W}_\alpha(\mathbf{x}_0)$ is the matrix of kriging weights (Myers, 1983; Goovaerts, 1997; Wackernagel, 2003; Chilès & Delfiner, 2012).

The derivation of the weights depends on the assumptions made about the global mean. As a result there are different kriging methods whose equations depend on these assumptions. Specifically one distinguishes:

- *Simple cokriging* (SCK). In this case the global mean is assumed to be constant. The weight matrix $\mathbf{W}_\alpha(\mathbf{x}_0)$ is derived from the *simple cokriging system*

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{C}(\mathbf{x}_\beta - \mathbf{x}_\alpha)\mathbf{W}_\alpha(\mathbf{x}_0) = \mathbf{C}(\mathbf{x}_\beta - \mathbf{x}_0).$$

- *Ordinary cokriging* (OCK). The global mean is usually unknown and so either approximated by the sample mean or, more commonly, the unknown mean is

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_6

filtered from the equations by imposing conditions on the sum of the weights. As a result, the estimator is replaced by the *ordinary cokriging estimator*, in which case the assumption of second order stationarity can be further weakened to intrinsic stationarity and the estimator can be expressed in terms of the variogram. The ordinary cokriging estimate $\boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0)$ at location $\mathbf{x}_0$ is given by

$$\boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0) = \sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha^{OCK}(\mathbf{x}_0)\boldsymbol{\zeta}(\mathbf{x}_\alpha)$$

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha^{OCK}(\mathbf{x}_0) = \mathbf{I}_{D-1},$$

where $\mathbf{W}_\alpha^{OCK}(\mathbf{x}_0)$ is the matrix of weights derived from the *ordinary cokriging system*

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \boldsymbol{\Gamma}(\mathbf{x}_\beta - \mathbf{x}_\alpha)\mathbf{W}_\alpha^{OCK}(\mathbf{x}_0) + \boldsymbol{\Lambda}(\mathbf{x}_0) = \boldsymbol{\Gamma}(\mathbf{x}_\beta - \mathbf{x}_0)$$

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha^{OCK}(\mathbf{x}_0) = \mathbf{I}_{D-1}.$$

The ordinary cokriging error covariance matrix is given by

$$\boldsymbol{\Sigma}_{OCK}(\mathbf{x}_0) = \sum_{\alpha=1}^{N(\mathbf{x}_0)} \boldsymbol{\Gamma}(\mathbf{x}_0 - \mathbf{x}_\beta)\mathbf{W}_\alpha^{OCK}(\mathbf{x}_0) - \boldsymbol{\Lambda}(\mathbf{x}_0), \tag{6.2}$$

and the *OCK estimation variance* may be written as

$$\sigma^2_{OCK}(\mathbf{x}_0) = \mathrm{Tr}\left[\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha^{OCK}(\mathbf{x}_0)\boldsymbol{\Gamma}(\mathbf{x}_0 - \mathbf{x}_\alpha)\right] - \mathrm{Tr}(\boldsymbol{\Lambda}(\mathbf{x}_0)).$$

The matrix $\boldsymbol{\Lambda}(\mathbf{x}_0)$ is the matrix of Lagrange multipliers.

- *Universal cokriging* (UCK). When the data exhibit a clear trend, *universal kriging* can be used to treat the trend as deterministic and the residuals as stochastic. It is assumed that the random function can be written as

$$\boldsymbol{\zeta}(\mathbf{x}) = \mathbf{m}(\mathbf{x}) + \mathbf{Y}(\mathbf{x}),$$

where $\mathbf{m}(\mathbf{x})$ denotes the deterministic component and $\mathbf{Y}(\mathbf{x})$ is a second order stationary random function. The deterministic component can be written as a linear combination of basis functions $f_i, i = 0, \ldots, k$ that can be polynomials,

sines and cosines or exponential functions of the coordinates.

$$\mathbf{m}(\mathbf{x}) = \mathbf{F}(\mathbf{x})\mathbf{a}.$$

Here, the ith column of $\mathbf{a}$ denotes the vector of drift coefficients for the random function ζ_i and $\mathbf{F}(\mathbf{x})$ denotes the row vector of basis functions evaluated at $\mathbf{x}$. Determination of the estimates requires solution of the universal kriging system

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \boldsymbol{\Gamma}(\mathbf{x}_\beta - \mathbf{x}_\alpha)\mathbf{W}_\alpha^{UCK}(\mathbf{x}_0) + \mathbf{F}(\mathbf{x}_\alpha)\boldsymbol{\Lambda}(\mathbf{x}_0) = \boldsymbol{\Gamma}(\mathbf{x}_\beta - \mathbf{x}_0)$$

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{F}(\mathbf{x}_\alpha)\mathbf{W}_\alpha^{UCK}(\mathbf{x}_0) = \mathbf{F}(\mathbf{x}_0).$$

The matrix $\boldsymbol{\Lambda}(\mathbf{x}_0)$ is the matrix of Lagrange multipliers.

The *UCK estimation variance* is given by

$$\sigma^2_{UCK}(\mathbf{x}_0) = \mathrm{Tr}\left[\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha^{UCK}(\mathbf{x}_0)\boldsymbol{\Gamma}(\mathbf{x}_0 - \mathbf{x}_\alpha)\right] - \mathrm{Tr}(\boldsymbol{\Lambda}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)).$$

6.2 Cokriging of the Mean

Instead of estimating the values at unsampled locations, cokriging can also be used to estimate the global mean in cases when the mean is unknown. To do so the OCK and UCK systems need to be modified as follows:

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \boldsymbol{\Gamma}(\mathbf{x}_\beta - \mathbf{x}_\alpha)\mathbf{W}_\alpha^{OCKm}(\mathbf{x}_0) + \boldsymbol{\Lambda}(\mathbf{x}_0) = \mathbf{0}$$

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{W}_\alpha^{OCKm}(\mathbf{x}_0) = \mathbf{I}_{D-1}$$

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \boldsymbol{\Gamma}(\mathbf{x}_\beta - \mathbf{x}_\alpha)\mathbf{W}_\alpha^{UCKm}(\mathbf{x}_0) + \mathbf{F}(\mathbf{x}_\alpha)\boldsymbol{\Lambda}(\mathbf{x}_0) = \mathbf{0}$$

$$\sum_{\alpha=1}^{N(\mathbf{x}_0)} \mathbf{F}(\mathbf{x}_\alpha)\mathbf{W}_\alpha^{UCKm}(\mathbf{x}_0) = \mathbf{F}(\mathbf{x}_0).$$

6.3 Comments

6.3.1 Implementation Practicalities

All three estimators make use of a moving neighbourhood $W(\mathbf{x}_0)$ that is typically circular or elliptical and assumed to contain $N(\mathbf{x}_0)$ sample locations. The neighbourhood is chosen based on the characteristics of the variogram model for the data and the type of estimator to be used.

For simple cokriging at least one sample has to lie within the search neighbourhood, for ordinary kriging the minimum number of samples is 2. For universal kriging the minimum depends on the number of functional constraints that are imposed. In order to avoid calculating estimates too far from the boundary of the sampled region it is common to set a higher minimum value. For estimates in 2D space it usually suffices to use a maximum of 20 samples in the estimation (Webster & Oliver, 2007).

If anisotropy is part of the characteristics of the data, then it is common practice to incorporate the anisotropy also in the search neighbourhood by orienting the search ellipse in the direction of greatest continuity and adjusting the lengths of the major and minor axis of the ellipse so that the anisotropy ratio is reflected.

Lastly if the contribution of the nugget component is large compared to the total sill of the variogram model, the number of samples included in the estimation needs to be raised in order to reflect the (relatively) poor continuity and this may require increasing the size of the search neighbourhood also.

In "`gmGeostats`", the neighbourhood can be defined by means of the function `KrigingNeighbourhood`. This offers a series of parameters, such as `nmax` and `nmin` (resp. maximum and minimum number of data in the neighbourhood), `omax` (maximum number of data per quadrant/octant) and `maxdist` (maximum distance between selected data and interpolation point). This creates an object of class "`KrigingNeighbourhood`" that can be passed to the cokriging function. In package "`gstat`", the strategy is somewhat different: here the arguments (with the same names) are passed to a call to `gstat` to attach the local neighbourhood definition to the resulting "`gstat`" object.

6.3.2 Properties

The above formulation of the cokriging estimators was based on alr-transformed data. One obvious objection to this method might be that there could be a dependence of the estimator on the choice of numerator for the alr-transform, or on another log-ratio transformation chosen to represent the data. However, this is not a problem, since it can be shown that the alr-cokriging estimator is invariant under permutations (Pawlowsky-Glahn & Olea, 2004).

Moreover, because of Eq. (4.11), we can move from the alr variogram back to the variation-variogram and hence by virtue of (4.10) to formulations in terms of the clr or the ilr transform. Actually, all cokriging estimators (and their covariances) defined above are affine equivariant (Tolosana-Delgado, 2006).

The cokriging estimates and cokriging covariance matrix may be used to define at each location $\mathbf{x}_0$ a local conditional distribution. This in turn allows an assessment of uncertainty about the composition at the location through the construction of confidence intervals. In addition the propagation of the uncertainty to further analyses is possible, such as likely benefit or cost. In these cases the assumption of the regionalised composition being a realisation of a multivariate Gaussian random function is required.

6.3.3 *Unbiased, in Which Scale?*

The estimates constructed through cokriging are unbiased in the space in which they were computed.

If alr-transformed data were used then, because of the relationships between the various log-ratio transforms, the estimates are unbiased for the other log-ratio transforms also.

This implies specifically that when back-transformed to the simplex, for example via agl, the estimates are unbiased with respect to the Aitchison geometry; however, this does not imply unbiasedness in the Euclidean geometry (Pawlowsky-Glahn & Olea, 2004).

The cokriging estimates are in alr-space and need to be back-transformed to compositions. An alternative method to the crude back-transform via the agl-transform is to apply *Gauss–Hermite quadrature* to compute an estimate of the expected value of the composition,

$$\boldsymbol{\mu}_{\mathbf{Z}}^{OCK}(\mathbf{x}_0) = \frac{1}{\sqrt{\pi^{D-1}}} \int_{\mathbb{R}^D} \mathrm{agl}(\sqrt{2}\mathbf{R}^t\mathbf{U} + \boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0)) \exp(-\mathbf{U}^t\mathbf{U}) d\mathbf{U}. \tag{6.3}$$

Here the matrix $\mathbf{R}$ denotes the Cholesky decomposition of $\boldsymbol{\Sigma}_{OCK}(\mathbf{x}_0)$. Setting $g_1(u_{i_1}, u_{i_2} \ldots, u_{i_D}) = \pi^{-(D-1)/2}\mathrm{agl}(\sqrt{2}\mathbf{R}^t\mathbf{U} + \boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0))$ the integral can be approximated numerically using k Gauss–Hermite quadrature points.

$$\boldsymbol{\mu}_{\mathbf{Z}}^{OCK}(\mathbf{x}_0) \approx \sum_{i_1=1}^{k} \sum_{i_2=1}^{k} \cdots \sum_{i_D=1}^{k} \prod_{\ell=1}^{D} w_{i_\ell} g_1(u_{i_1}, u_{i_2}, \ldots, u_{i_D}).$$

To obtain an expression for the variance–covariance matrix in the simplex we put

$$\boldsymbol{\Sigma}_Z(\mathbf{x}_0) = \frac{1}{\sqrt{\pi^{D-1}}} \int_{\mathbb{R}^D} g_2(\mathbf{U}) \exp(-\mathbf{U}^t\mathbf{U}) d\mathbf{U},$$

where

$$g_2(\mathbf{U}) = (\text{agl}(\sqrt{2}\mathbf{R}^t\mathbf{U} + \boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0)) - \boldsymbol{\mu}_{\mathbf{Z}}^{COK}(\mathbf{x}_0))(\text{agl}(\sqrt{2}\mathbf{R}^t\mathbf{U} + \boldsymbol{\zeta}^*_{OCK}(\mathbf{x}_0)) - \boldsymbol{\mu}_{\mathbf{Z}}^{OCK}(\mathbf{x}_0))^t$$

and just as in the case of the mean, the integral can be approximated via Gauss–Hermite quadrature.

6.3.4 Cokriging in R

Several packages provide functionality to perform the various variants of cokriging, notably "`gmGeostats`", "`gstat`", "`geoR`" (Diggle & Ribeiro, 2007), "`RandomFields`" (Schlather et al., 2020) and "`RGeostats`" (MINES ParisTech/ARMINES, 2019). The package "`gmGeostats`" offers its own functionality, as well as convenience functions bridging several of these packages.

As explained in Chaps. 4 and 5, both packages "`gmGeostats`" and "`gstat`" require the establishment of a data container, respectively, with functions `make.gmCompositionalGaussianSpatialModel` and `gstat`, which have so far been used to construct variogram models and LMCs for the data (compositional in the first case, generally multivariate in the second case).

The commands also provide for the specification of the kriging estimator, via the `formula` term, which for the ordinary cokriging is `~1`, while for universal kriging or kriging with a trend a function of the coordinates needs to be supplied, e.g. `~ Easting+Northing` or `~trendVariable`.

In addition, a search neighbourhood and the variogram model need to be specified.

6.4 Cokriging Estimation of Windarling Data

6.4.1 Ordinary Cokriging

To compute cokriging estimates we will first define a grid of locations at which estimates are to be made. Given the sample separation, the mesh size will be set to 1m by 1m, and it will cover the extent of the two variables where we do have data

```
> xmin=floor(min(wind.coords[,1]))
> xmax=ceiling(max(wind.coords[,1]))
> ymin=floor(min(wind.coords[,2]))
> ymax=ceiling(max(wind.coords[,2]))
> x0 = c(xmin, ymin)
> names(x0) = colnames(wind.coords)
> Dx = c(1,1)
> nx = c(xmax-xmin, ymax-ymin)/Dx +1
```

With these parameters, we can construct the grid topology, and the grid itself

```
> (wind.gt = GridTopology(x0, Dx, nx))
```

```
                  Easting Northing
cellcentre.offset    -236       15
cellsize                1        1
cells.dim             442      109
```

```
> wind.grid.fine = SpatialGrid(wind.gt)
```

The OCK estimates are now computed using `wind.pwlr.mdAnis`, the anisotropic LMC constructed via MAF on page 101, Fig. 5.7, Chap. 5. OCK will be performed in alr-coordinates and a maximum of 20 neighbouring samples will be used in the estimation

```
> wind.ng = KrigingNeighbourhood(nmax=20, nmin=4, maxdist=20)
```

The use of OCK is flagged via `formula`, which in this case is `"∼ 1"`. We also check the names of the output columns to ascertain the ordering in which the output from OCK is stored. Inclusion of debug.level $= -1$ allows one to monitor progress of the estimation.

```
> wind.acomp=acomp(wind.compo)
> wind.alr.ggAnis = make.gmCompositionalGaussianSpatialModel(
+   wind.compo, wind.coords, V="alr", ng = wind.ng,
+   formula = ~1, model=wind.pwlr.mdAnis)
> wind.alr.ock = predict(wind.alr.ggAnis, newdata=wind.grid.fine)
```

```
> class(wind.alr.ock)
```

```
[1] "SpatialGridDataFrame"
attr(,"package")
[1] "sp"
```

```
> names(wind.alr.ock@data)
```

```
 [1] "alr1.pred"      "alr1.var"       "alr2.pred"
 [4] "alr2.var"       "alr3.pred"      "alr3.var"
 [7] "alr4.pred"      "alr4.var"       "alr5.pred"
[10] "alr5.var"       "cov.alr1.alr2"  "cov.alr1.alr3"
[13] "cov.alr2.alr3"  "cov.alr1.alr4"  "cov.alr2.alr4"
[16] "cov.alr3.alr4"  "cov.alr1.alr5"  "cov.alr2.alr5"
[19] "cov.alr3.alr5"  "cov.alr4.alr5"
```

The estimates and estimation error variance for $\zeta_1 = \log(\frac{Fe}{R})$ are first plotted to check the output for plausibility. As is expected the cokriging variance is low near sample locations and increases with increasing distance from conditioning locations. From the map of the cokriging variance it is also clear that some extrapolation has taken place (Fig. 6.1).

A mask is therefore required to suppress extrapolations. Several approaches are possible. One of these is to define the mask via a percentage of the total sills of all variances as a reference, for example, 90%, something done automatically by the function `constructMask`. In the masked output for $\zeta_1 = \log(\frac{Fe}{R})$ the shape of the region as indicated by the conditioning data is well reproduced (Fig. 6.2)

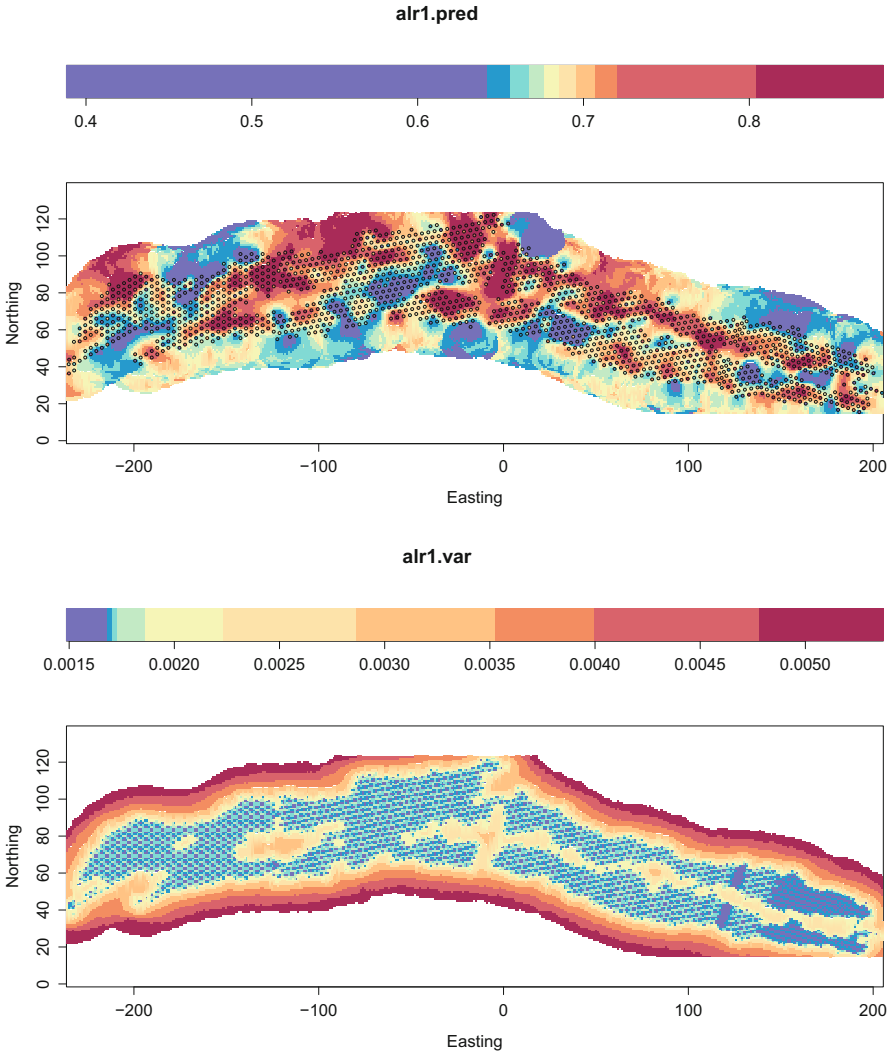

Fig. 6.1 Cokriging estimate (top) and estimation variance (bottom) of $\log(\frac{Fe}{R})$

```
> aux = constructMask(wind.alr.ock, maxval= 0.9,
+                     method="sillprop", x=wind.alr.gg)
> image(aux)
```

An alternative is to define a mask based on the shape of the region and the distance from the sample data. This can be done using the same function `constructMask` with extra argument `method="maxdist"`, which internally makes use of the

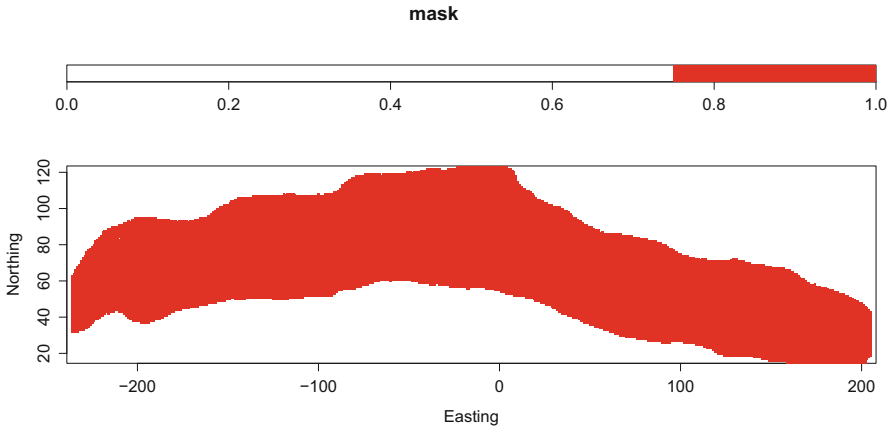

Fig. 6.2 OCK estimation region, estimation variances of $\log(\frac{Fe}{R})$ above 0.9 masked

package "FNN" and then set the values outside the region to NA

```
> wind.mask.fine = constructMask(wind.grid.fine, maxval=7,
+                                method="maxdist",x=wind.alr.ggAnis)
> wind.alr.ock[!wind.mask.fine,] <- NA
```

The auxiliary function `constructMask` admits several algorithms to generate the mask, including the distance based one (`method="maxdist"`) and the variogram sill one (`method="sillprop"`), as well as a "`SpatialPolygon`"-based one (`method="point2poly"`).

The estimates in alr-coordinates are consistent with the input data, so we can proceed to back-transform the estimates to compositions and assess the quality of the estimates in raw data space. The assessment will include visualisation and summary statistics. Maps of the estimates can be obtained via

```
> wind.compo.ock = gsi.gstatCokriging2compo(
+         wind.alr.ock, V="alr", orignames=colnames(wind.compo))
```

Maps of the estimates in Fig. 6.3 show good coincidence with the sample data. We can visualise this with function `image_cokriged`, which invisibly returns the breaks and colour scale to use for further plotting, for instance, adding points filled with colour using the same legend

```
> myfun = function(i){
+   bks = quantile(wind.acomp[,i], probs=seq(0,1,0.1), na.rm=T)
+   out = image_cokriged(wind.compo.ock, ivar=i, breaks=bks)
+   points(wind.coords, pch=21, col=1, cex=0.5,
+          bg=out$col[cut(wind.acomp[,i],out$breaks)])
+ }

> sapply(c("Fe", "Al2O3", "SiO2", "Mn", "P", "R"), myfun)
```

Once a mask is available, calculations can be constrained to the region within the mask. To skip calculations for points outside of the region defined by the mask we

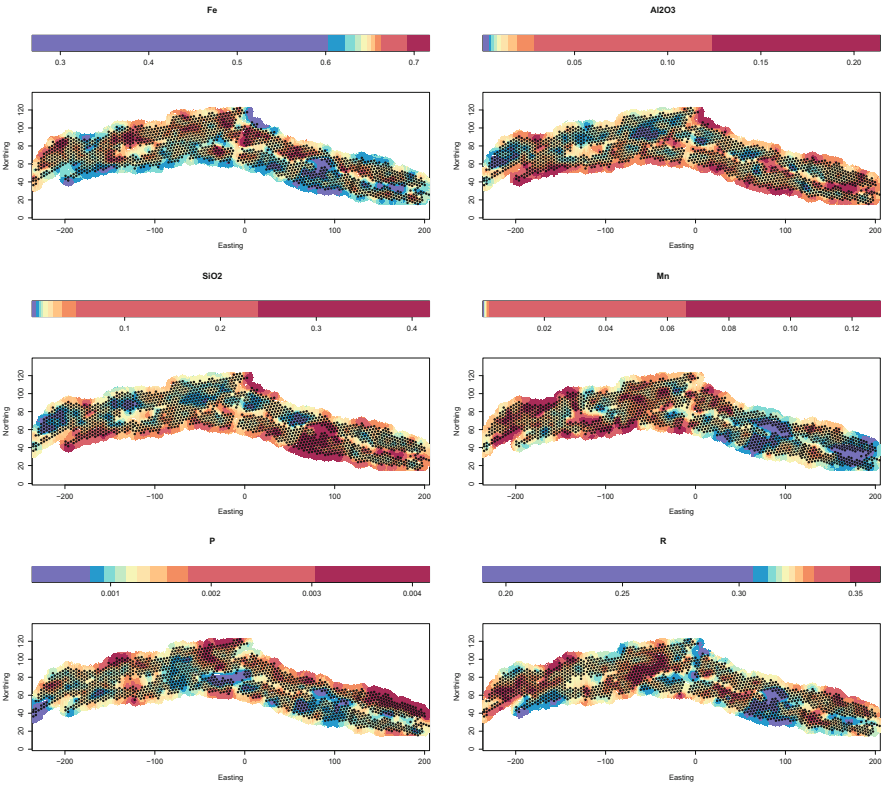

Fig. 6.3 OCK estimates for Windarling data with model `wind.alr.ggAnis`

will illustrate here the use of functions `setMask` and `unmask`. In the case of the OCK estimates this may be done as follows:

```
> wind.grid.fine.masked = setMask(wind.grid.fine, wind.mask.fine)
> wind.alr.ock = predict(wind.alr.ggAnis,
+                        newdata=wind.grid.fine.masked)%>%
+                                  unmask(mask=wind.mask.fine)

starting cokriging
Linear Model of Coregionalization found. Good.
[using ordinary cokriging]

> wind.compo.ock = gsi.gstatCokriging2compo(wind.alr.ock,
+                   V="alr", orignames = colnames(wind.compo))
> wind.alr.ock.trend = predict(wind.alr.ggAnis,
+                  newdata=wind.grid.fine.masked, BLUE=TRUE) %>%
+   unmask(mask=wind.mask.fine)

starting cokriging
Linear Model of Coregionalization found. Good.
[generalized least squares trend estimation]
```

```
> wind.compo.ock.trend =
+   gsi.gstatCokriging2compo(wind.alr.ock.trend, V="alr",
+                                orignames = colnames(wind.compo))
```

First all points of the grid `wind.grid.fine` are masked that lie in the exterior of the mask `wind.mask.fine`. Then OCK estimates are calculated at each point of the resulting `wind.grid.fine.masked`, and the estimates are automatically unmasked with the command `unmask(x, mask)`. These can then be back-transformed to compositional estimates with `gsi.gstatCokriging2compo` as before.

6.4.2 Universal Cokriging

The spatial analysis in Chap. 3 suggested that the ordinary cokriging with an anisotropic LMC might not be the most appropriate type of kriging to use. Given that the swath plots and the variograms indicate a trend in the NS direction, universal kriging, with a trend on the NS direction and an isotropic LMC, might be an appropriate alternative model. This is particularly straightforward to do using "`gstat`"-variograms, because the anisotropy is specified in these variogram models by means of just anisotropy ratios. Hence, a new "`gstatVariogram`" object is defined, using the model defined in the object `wind.alr.ggAnis`, and the anisotropy ratio of all components is set to one

```
> aux = wind.alr.ggAnis %>% as.gstat
> for (i in 1:length(aux$model)){
+   aux$model[[i]]$anis1=1}
> wind.pwlr.md = as.LMCAnisCompo(aux$model, V="alr")
```

Then we feed that new isotropic model to the geospatial object

```
> wind.alr.gg.uck = make.gmCompositionalGaussianSpatialModel(
+   data=wind.acomp, coords=wind.coords, V="alr", ng=wind.ng,
+   formula = ~1+Northing, model = wind.pwlr.md)
```

and proceed with prediction, unmasking and recasting to compositions as shown before

```
> wind.alr.uck =
+   predict(wind.alr.gg.uck, newdata=wind.grid.fine.masked) %>%
+         unmask(wind.mask.fine)
> wind.compo.uck = gsi.gstatCokriging2compo(wind.alr.uck, V="alr",
+                                  orignames=colnames(wind.compo))
```

As a first step, the estimates of the alr variables are plotted. A spatial map of the estimates for $\zeta_1 = \log(\frac{Fe}{R})$ is provided in Fig. 6.4

```
> image_cokriged(wind.alr.uck, "alr1.pred")
```

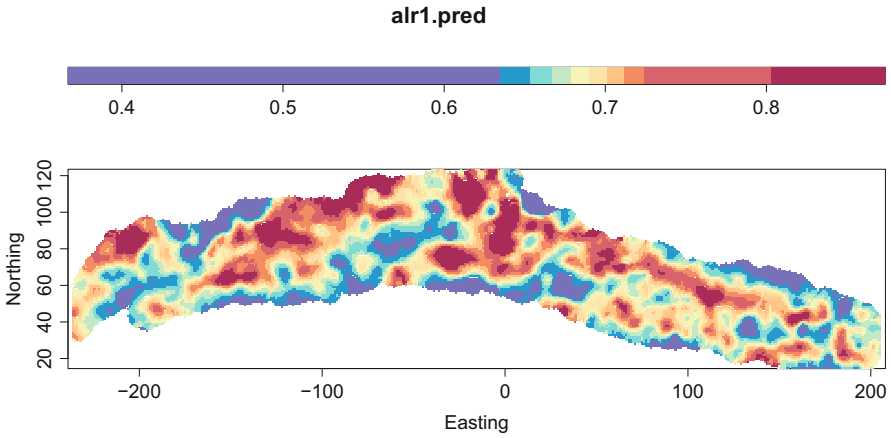

Fig. 6.4 Spatial map of UCK estimates of $\zeta_1 = \log(\frac{Fe}{R})$

The estimates in the raw compositional scale are shown in Fig. 6.5, created with this chunk:

```
> myfun = function(i){
+   bks = quantile(wind.acomp[,i], probs=seq(0,1,0.1), na.rm=T)
+   image_cokriged(wind.compo.uck, ivar=i, breaks=bks)
+ }

> sapply(c("Fe", "Al2O3", "SiO2", "Mn", "P", "R"), myfun)
```

We can then plot it with a comparison with the original data. This is easily available using the function `image_cokriged`, which has a method for objects of class "`spatialGridAcomp`". This invisibly returns the breaks and colour regions applied to the values of the displayed variable, and a plot of the UCK estimates of Fe with the sample data is shown in Fig. 6.6

```
> bks = quantile(wind.acomp[,"Fe"], probs=seq(0,1,0.1), na.rm=T)
> dsc = image_cokriged(wind.compo.uck,ivar="Fe",breaks=bks)
> points(wind.coords, cex=0.5, col="black", pch=22,
+        bg=dsc$col[cut(wind.acomp[,"Fe"], breaks=dsc$breaks)])
```

Equivalent diagrams for the other components can be constructed similarly.

6.4.3 *Estimation of the Local and Global Mean*

To estimate the local mean the same commands as for OCK and UCK can be used with the addition of the flag `BLUE=TRUE`

```
> wind.alr.ock.trend = predict(wind.alr.ggAnis,
+                   newdata=wind.grid.fine.masked, BLUE=TRUE) %>%
+   unmask(mask=wind.mask.fine)
```

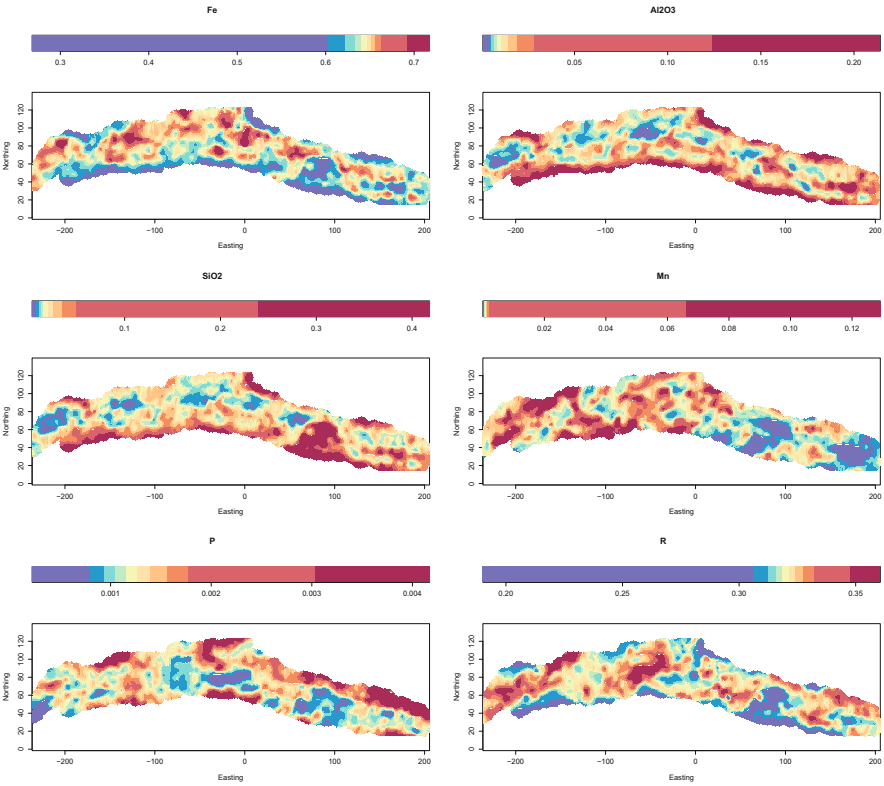

Fig. 6.5 UCK estimates for Windarling data with model `wind.alr.gg.uck`

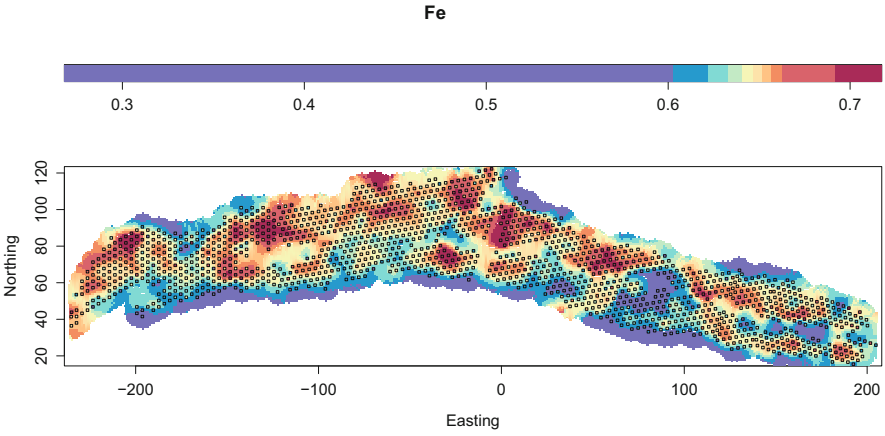

Fig. 6.6 Spatial map of UCK estimates of Fe with sample data superimposed

```
starting cokriging
Linear Model of Coregionalization found. Good.
[generalized least squares trend estimation]
```

```
> wind.compo.ock.trend =
+   gsi.gstatCokriging2compo(wind.alr.ock.trend, V="alr",
+                            orignames = colnames(wind.compo))
> wind.alr.uck.trend = predict(wind.alr.gg.uck, BLUE=TRUE,
+                           newdata=wind.grid.fine.masked) %>%
+     unmask(wind.mask.fine,wind.grid.fine)
```

```
starting cokriging
Linear Model of Coregionalization found. Good.
[generalized least squares trend estimation]
```

```
> wind.compo.uck.trend =
+   gsi.gstatCokriging2compo(wind.alr.uck.trend, V="alr",
+                            orignames = colnames(wind.compo))
```

```
> names(wind.compo.ock.trend)
```

```
[1] "Fe"    "Al2O3" "SiO2"  "Mn"    "P"     "R"
```

```
> names(wind.compo.uck.trend)
```

```
[1] "Fe"    "Al2O3" "SiO2"  "Mn"    "P"     "R"
```

Images of the local mean estimates for Fe and P with OCK and UCK are shown in the first and second rows of Fig. 6.7. These are created with chunks such as

```
> bks = quantile(wind.acomp[,"Fe"], probs=seq(0,1,0.1), na.rm=T)
> out = image_cokriged(wind.compo.ock.trend,
>                                          ivar="Fe", breaks=bks)
> points(wind.coords, pch=21, col=1, cex=0.5,
+          bg=out$col[cut(wind.acomp[,"Fe"],out$breaks)])
> mtext(side=1, outer=T, text = "local OCK")
```

```
> bks = quantile(wind.acomp[,"P"], probs=seq(0,1,0.1), na.rm=T)
> out = image_cokriged(wind.compo.ock.trend,
>                                          ivar="P", breaks=bks)
```

making use of the fact that the function `image_cokriged` for objects of class "`spatialGridAcomp`" invisibly returns the breaks and colour regions applied to the values of the displayed variable. The other rows show the output of these trend calculations for the same two variables, but in the case of considering global neighbourhoods, i.e. with the following objects (note the absence of a neighbourhood object)

```
> wind.alr.ggAnis.Global =
+  make.gmCompositionalGaussianSpatialModel(
+    data=wind.acomp, coords=wind.coords, V="alr",
+    formula = ~1, model=wind.pwlr.mdAnis)
> wind.alr.gg.uck.Global =
+   make.gmCompositionalGaussianSpatialModel(
+    data=wind.acomp, coords=wind.coords, V="alr",
+    formula = ~1+Northing, model=wind.pwlr.md)
```

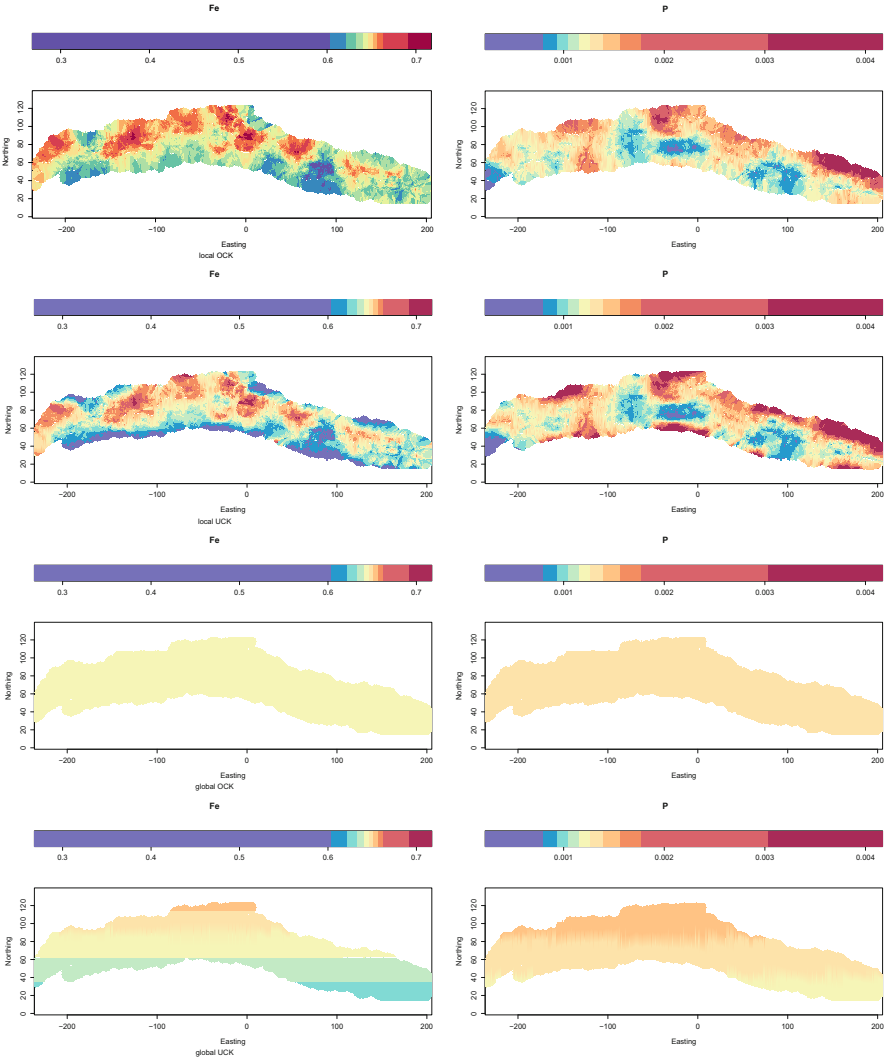

Fig. 6.7 Local (top) and global (bottom) trend estimates for Fe (left) and P (right) at Windarling with anisotropic ordinary cokriging and with isotropic universal kriging with NS trend. Note that each variable has its own colour scale

Generally the local means follow the patterns in the OCK and UCK estimates but the maps are smoother than those of the OCK estimates. Readers are invited to make these maps for each variable, and check that for OCK the global means are always constant, while for UCK we see an increase from the south to the north for Fe, Mn, P and the Rest. For Al_2O_3 and SiO_2 there is an increase from the north to the south. These trends are consistent with what we noted on the spatial maps earlier on.

6.4.4 Comparison of OCK and UCK Results

Having considered spatial maps of the estimates the summary statistics of the OCK and UCK estimates need to be considered both in the raw data scales as well as from a compositional perspective.

```
> wind.compo.ock %>% setMask(mask=wind.mask.fine) %>% summary()
```

```
       Fe              Al2O3               SiO2          
 Min.   :0.443   Min.   :0.00145   Min.   :0.00508  
 1st Qu.:0.621   1st Qu.:0.00763   1st Qu.:0.01398  
 Median :0.640   Median :0.01237   Median :0.02131  
 Mean   :0.634   Mean   :0.01530   Mean   :0.02862  
 3rd Qu.:0.652   3rd Qu.:0.02019   3rd Qu.:0.03416  
 Max.   :0.691   Max.   :0.10745   Max.   :0.28404  
       Mn                 P                  R          
 Min.   :0.000053   Min.   :0.000257   Min.   :0.235  
 1st Qu.:0.000556   1st Qu.:0.001014   1st Qu.:0.315  
 Median :0.000946   Median :0.001226   Median :0.321  
 Mean   :0.001342   Mean   :0.001280   Mean   :0.320  
 3rd Qu.:0.001532   3rd Qu.:0.001499   3rd Qu.:0.326  
 Max.   :0.027643   Max.   :0.003155   Max.   :0.348  
```

```
> wind.compo.uck %>% setMask(mask=wind.mask.fine) %>% summary()
```

```
       Fe              Al2O3               SiO2          
 Min.   :0.282   Min.   :0.00147   Min.   :0.00521  
 1st Qu.:0.616   1st Qu.:0.00772   1st Qu.:0.01418  
 Median :0.639   Median :0.01292   Median :0.02179  
 Mean   :0.628   Mean   :0.01898   Mean   :0.03214  
 3rd Qu.:0.651   3rd Qu.:0.02218   3rd Qu.:0.03745  
 Max.   :0.690   Max.   :0.24796   Max.   :0.30862  
       Mn                 P                  R          
 Min.   :0.00005   Min.   :0.000256   Min.   :0.185  
 1st Qu.:0.00055   1st Qu.:0.001009   1st Qu.:0.313  
 Median :0.00093   Median :0.001242   Median :0.320  
 Mean   :0.00149   Mean   :0.001324   Mean   :0.318  
 3rd Qu.:0.00159   3rd Qu.:0.001538   3rd Qu.:0.326  
 Max.   :0.05885   Max.   :0.004598   Max.   :0.347  
```

```
> wind.acomp %>% unclass() %>% summary()
```

```
       Fe              Al2O3               SiO2          
 Min.   :0.268   Min.   :0.0008   Min.   :0.0034  
 1st Qu.:0.623   1st Qu.:0.0054   1st Qu.:0.0106  
 Median :0.644   Median :0.0095   Median :0.0175  
 Mean   :0.633   Mean   :0.0154   Mean   :0.0297  
 3rd Qu.:0.657   3rd Qu.:0.0186   3rd Qu.:0.0333  
 Max.   :0.717   Max.   :0.2141   Max.   :0.4182  
       Mn                 P                  R          
 Min.   :0.00003   Min.   :0.00022   Min.   :0.190  
 1st Qu.:0.00049   1st Qu.:0.00094   1st Qu.:0.312  
 Median :0.00085   Median :0.00121   Median :0.320  
 Mean   :0.00189   Mean   :0.00128   Mean   :0.318  
```

```
 3rd Qu.:0.00150   3rd Qu.:0.00156   3rd Qu.:0.327
 Max.   :0.12918   Max.   :0.00417   Max.   :0.361
```

Plots of kernel density estimates of the OCK and UCK estimate distribution together with those of the conditioning data are shown in Fig. 6.8. The kernel densities of the UCK estimates are closer in shape to those of the raw data than the OCK estimates. Boxplots shown in Fig. 6.9 also confirm the longer tails in the estimates based on UCK compared to OCK and highlight the smoothing effect of the OCK estimator with ranges for estimates being lower than those of the raw data.

```
> par(mfrow=c(2,3))
> for (i in 1:6){
+ boxplot(wind.compo[,i],
+          setMask(wind.compo.ock, wind.mask.fine)[,i],
+          setMask(wind.compo.ock, wind.mask.fine)[,i],
+          horizontal=T, names=c("Sample","OCK","UCK"),
+          main=colnames(wind.compo)[i])
+ }
```

```
> par(mfrow=c(2,3))
> for (i in colnames(wind.compo)){
+   kde = list()
+   ock = setMask(wind.compo.ock, wind.mask.fine)[,i]
+   kde[["ock"]] = density(ock)
+   uck = setMask(wind.compo.uck, wind.mask.fine)[,i]
+   kde[["uck"]] = density(uck)
+   kde[["data"]] = density(wind.compo[,i])
+   xlim = range(sapply(kde, function(x) x$x))
+   ylim = range(sapply(kde, function(x) x$y))
+   plot(kde[["data"]], col="black", xlim=xlim,
+                                    ylim=ylim, main=i)
+   lines(kde[["ock"]], col="red")
+   lines(kde[["uck"]], col="blue")
+ }
```

The observations from the density plots and the boxplots are also confirmed by QQ-plots shown in Fig. 6.10.

```
> par(mfrow=c(6,3))
> for (i in 1:6){
+ x=quantile(wind.compo[,i],probs = seq(0,1,0.01))
+ y1=quantile(setMask(wind.compo.ock, wind.mask.fine)[,i],
+              probs = seq(0,1,0.01),na.rm=T)
+ y2=quantile(setMask(wind.compo.uck, wind.mask.fine)[,i],
+              probs = seq(0,1,0.01),na.rm=T)
+ plot(x,y1,xlab="true",ylab="OCK-predicted",asp=1,
+      main=names(wind.compo[i]))
+ abline(a=0,b=1)
+ plot(x,y2,xlab="true",ylab="UCK-predicted",asp=1,
+      main=names(wind.compo[i]))
+ abline(a=0,b=1)
+ plot(y1,y2,xlab="OCK-predicted",ylab="UCK-predicted",asp=1,
+      main=names(wind.compo[i]))
```

```
+ abline(a=0,b=1)
+ }
```

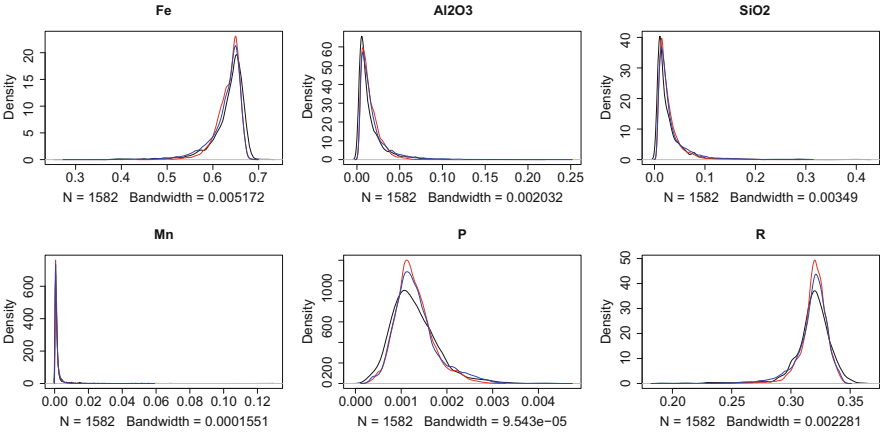

Fig. 6.8 Kernel density estimates of OCK (red) and UCK (blue) estimates and raw data (black)

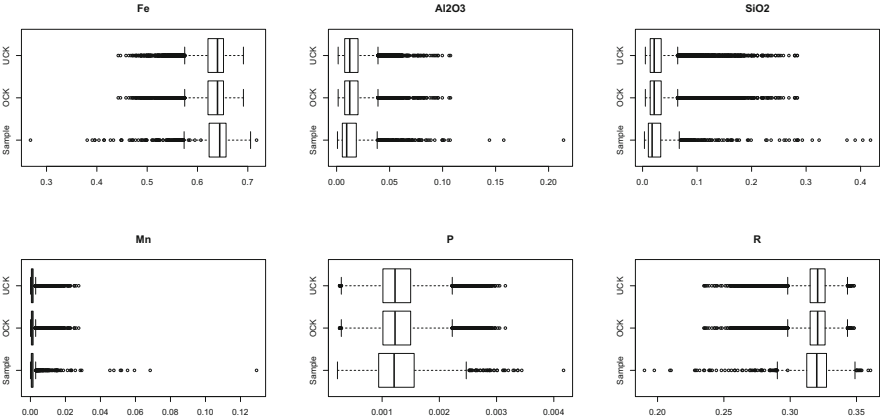

Fig. 6.9 Boxplots comparing conditioning data, OCK and UCK estimates

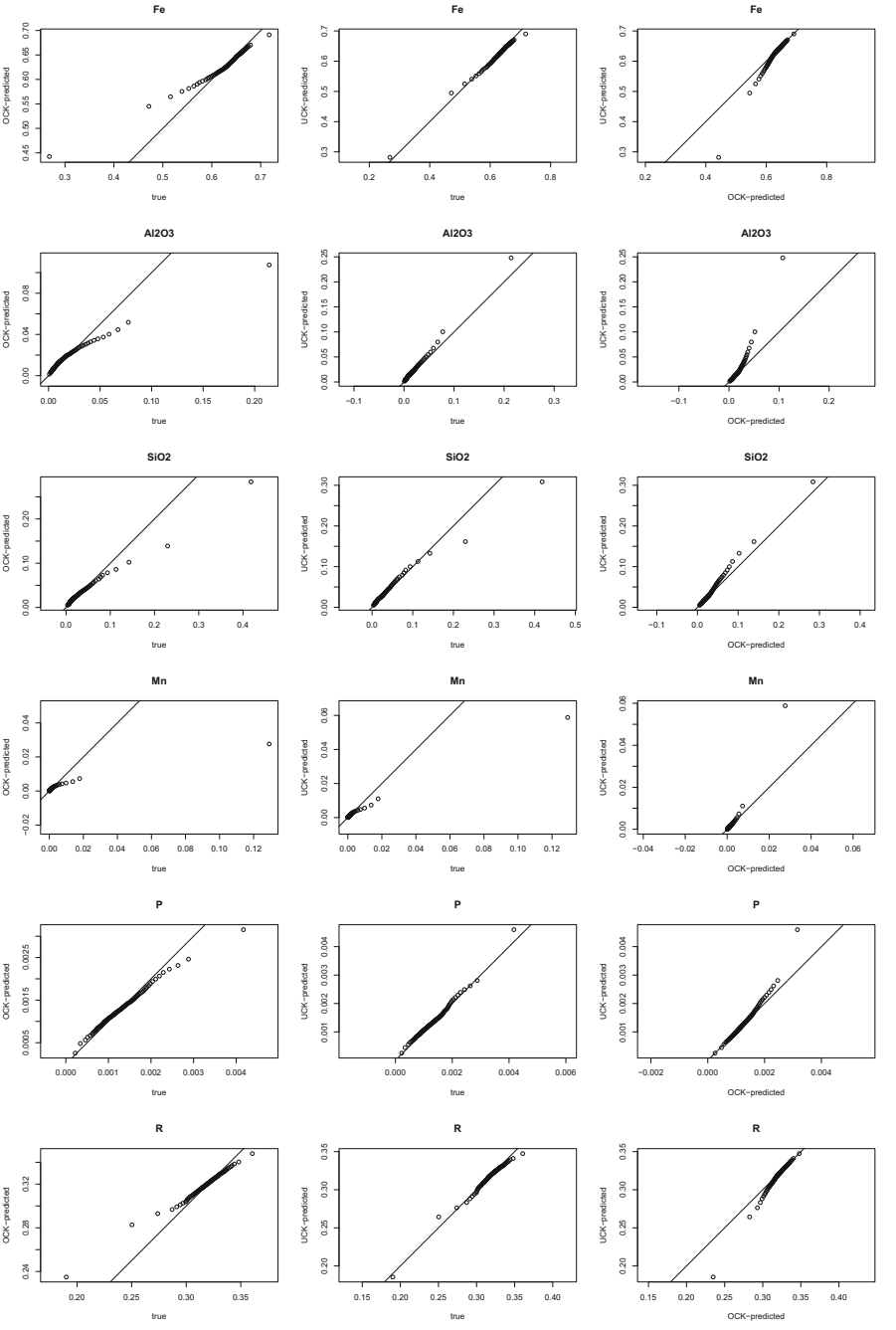

Fig. 6.10 QQ-plots of OCK estimates against conditioning data (left), UCK estimates against conditioning data (centre) and OCK estimates against UCK estimates (right)

Biplots of the estimates are shown in Fig. 6.11 and relevant ternary diagrams in Fig. 6.12.

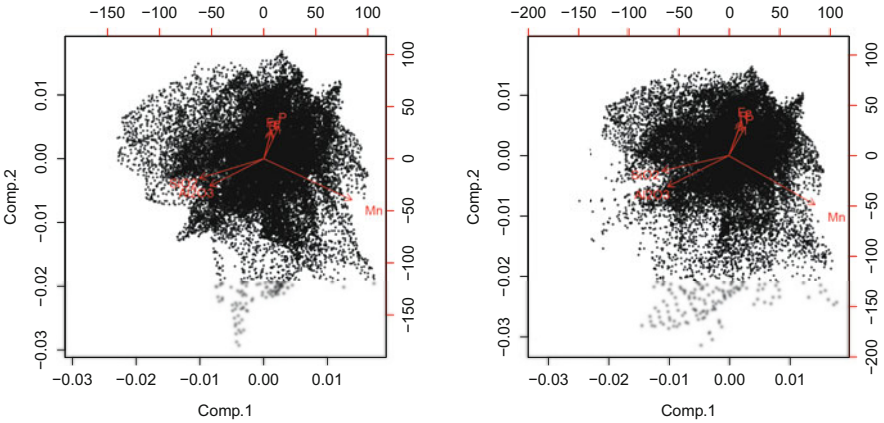

Fig. 6.11 Biplots for OCK (left) and UCK (right)

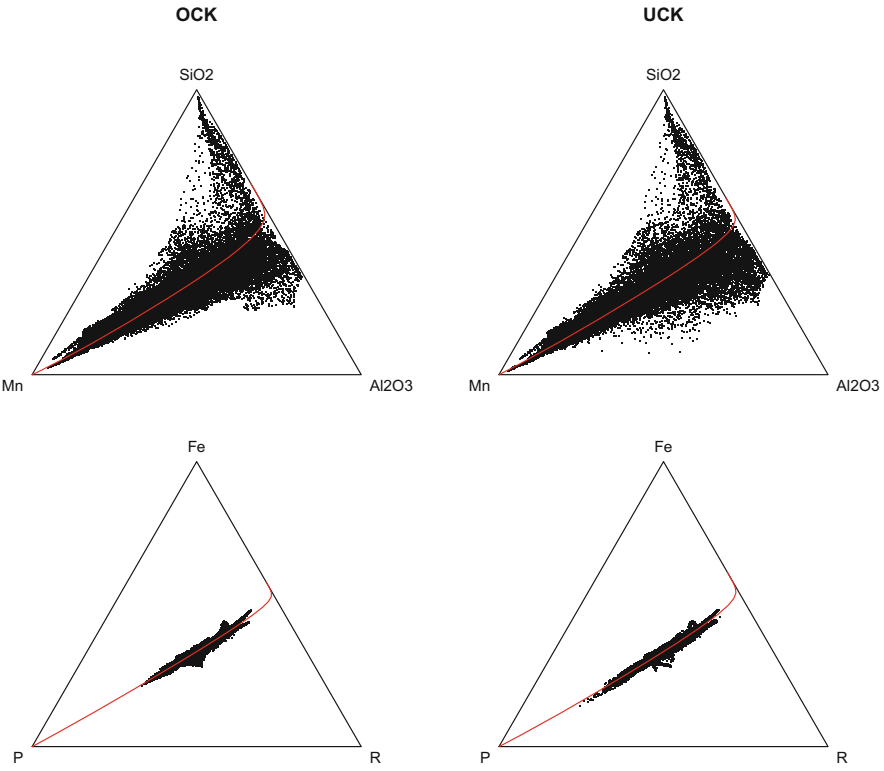

Fig. 6.12 Distributions of predictions with OCK and UCK on ternary diagram. Compare with those of Fig. 3.8

These two figures are made with the following code:

```
> par(mfcol=c(1,2))
> OCK.pc = setMask(wind.compo.ock, wind.mask.fine) %>% clr %>%
+   princomp
> coloredBiplot(OCK.pc, choices = 1:2, xlabs.pc=1, cex=c(0.25,1))
> UCK.pc = setMask(wind.compo.uck, wind.mask.fine) %>% clr %>%
+   princomp
> coloredBiplot(UCK.pc, choices = 1:2, xlabs.pc=1, cex=c(0.25,1))
```

```
> par(mfcol=c(2,2), mar=c(4,4,2,4))
> subcompo1 =c("Mn", "Al2O3", "SiO2")
> subcompo2 = c("P", "R", "Fe")
> wind.compo.ock[,subcompo1] %>% na.omit %>% acomp %>%
+   plot(center=T, pca=T, col.pca=2, pch=".")
> title(main="OCK")
> wind.compo.ock[,subcompo2] %>% na.omit %>% acomp %>%
+   plot(center=T, pca=T, col.pca=2, pch=".")
> wind.compo.uck[,subcompo1] %>%  na.omit %>% acomp %>%
+   plot(center=T, pca=T, col.pca=2, pch=".")
> title(main="UCK")
> wind.compo.uck[,subcompo2]  %>% na.omit %>% acomp %>%
+   plot(center=T, pca=T, col.pca=2, pch=".")
```

6.5 Estimation of Tellus Data Subcomposition

In Chap. 4 we had computed the MAF factors for the subcomposition MgO, Al_2O_3, CaO, Fe_2O_3 and R within the subset Tellus.subset. The composition was further analysed using compositional MAF and as a result we have four MAF factors, which based on Figs. 4.22, 4.23 exhibit negligible spatial cross-correlation. We can therefore use the mafs to provide estimates of the spatial distribution of the subcomposition.

As a first step the fitting of variogram models is required.

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

From the variation-variogram maps in Fig. 6.13 it is clear that there is some anisotropy in the direction N135. This is largely inherited by the first MAF factor.

```
> tellus.pwlr.vgAnis = logratioVariogram(
+   data=acomp(tellus.compo[tellus.subset,]),
+   loc=tellus.coords[tellus.subset,], maxdist=100000, nbins=10,
+   azimuth.tol=22.5, azimuth=10*(0:35))
> image(tellus.pwlr.vgAnis)
```

For each MAF factor a new "gstat" object will be constructed, together with a suitable variogram model and search parameters for the kriging estimation. An exemplar is shown below for maf1. The remaining "gstat" objects are generated in a similar fashion.

```
> tellusS.maf1.gg = gstat(id="maf1", formula=maf1~1,
+                     locations = ~EASTING+NORTHING,
```

```
+                    data=cbind(tellus.coords[tellus.subset,],
+                    maf.tellusS$scores))
> tellusS.maf1.vg = variogram(tellusS.maf1.gg, cutoff=100000,
+                             width=10000,
+                             alpha=c(45,135) )
> tellusS.maf1.md = vgm(add.to=vgm(model="Exp", nugget=0.35,
+                          psill=.9, range=70000,anis=c(135,.7)),
+                          model="Exp",range=120000, psill=0.1,
+                         anis=c(135,.9) )
> tellusS.maf1.gg = gstat(id="maf1", formula=maf1~1,
+                     locations = ~EASTING+NORTHING,
+                     data=cbind(tellus.coords[tellus.subset,],
+                     maf.tellusS$scores),
+                     model=tellusS.maf1.md,
+                     nmax=20,nmin=4,maxdist=60000)
```

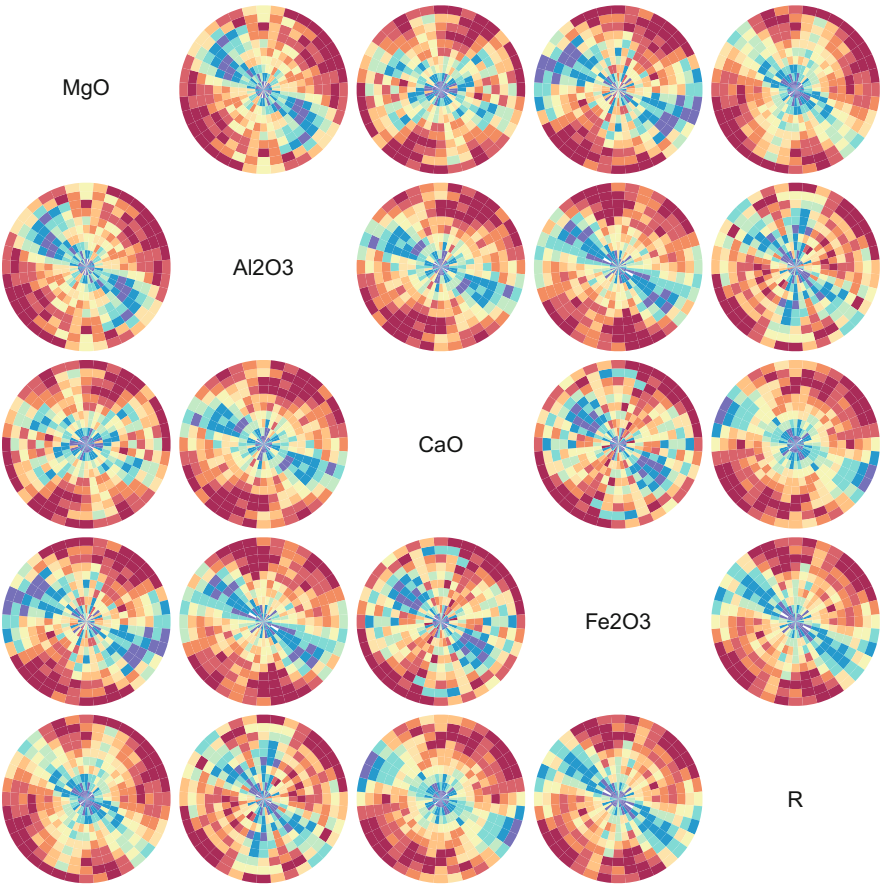

Fig. 6.13 Variation-variogram maps for the Tellus subcomposition in the sample set subset

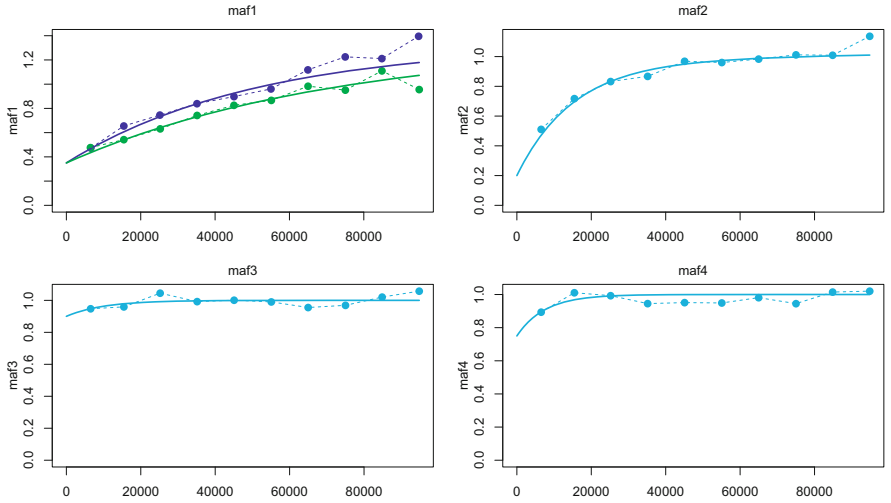

Fig. 6.14 Variogram models together with experimental variograms for maf1 to maf4

Plots of the variogram models for the MAF factors together with the experimental semivariograms are shown in Fig. 6.14.

Now that variogram models for the MAF factors have been constructed, we define an estimation grid and a mask to constrain the estimation region to Northern Ireland. In this instance we base the mask on the full Tellus sample locations.

```
> cellSize = 1400
> x = seq(min (tellus.coords [,1]) - cellSize/2,
+         max (tellus.coords [,1]) + cellSize, cellSize/2)
> y= seq (min (tellus.coords [,2]) - cellSize/2,
+         max (tellus.coords [,2]) + cellSize,cellSize/2)
> tellus.grid <- expand.grid (x , y )
> colnames(tellus.grid)=colnames(tellus.coords)
> tellus.mask=constructMask(tellus.grid,method = "maxdist",
+            maxval = 1400, x=cbind(tellus.coords,tellus.compo))
> tellus.grid.masked=setMask(tellus.grid, tellus.mask)
```

Then the estimates are calculated and because of the spatial decorrelation evident in the mafs, it suffices to use ordinary kriging. Here we show the process for the scores of the first and the last mafs (the rest are done in the same way).

```
> tellusS.maf1.ok = predict(tellusS.maf1.gg,
+                           newdata=tellus.grid.masked)

[using ordinary kriging]

> tellusS.maf1.ok=unmask(tellusS.maf1.ok, mask=tellus.mask)
> tellusS.maf4.ok = predict(tellusS.maf4.gg,
+                           newdata=tellus.grid.masked)

[using ordinary kriging]
```

```
> tellusS.maf4.ok=unmask(tellusS.maf4.ok,tellus.mask)

[using ordinary kriging]

[using ordinary kriging]
```

The spatial map for the first MAF factor is shown in Fig. 6.15

```
> out=tellusS.maf1.ok %>% image_cokriged(ivar="maf1.pred",
+                                        legendPos = "right")
> points(tellus.coords[tellus.subset,], pch=21, col=1,
+          bg=out$col[cut(maf.tellusS$scores[,1],out$breaks)])
```

The OK estimates of the MAF factors are now back-transformed to the raw data space to recover the original subcomposition.

```
> aux1=unlist(as.matrix(tellusS.maf1.ok[,3]))
> aux2=unlist(as.matrix(tellusS.maf2.ok[,3]))
> aux3=unlist(as.matrix(tellusS.maf3.ok[,3]))
> aux4=unlist(as.matrix(tellusS.maf4.ok[,3]))
> aux=cbind(aux1,aux2,aux3,aux4)
> tellusS.compo.ok=predict(maf.tellusS,aux)
> tellusS.compo.ok=cbind(tellus.grid,tellusS.compo.ok)
> colnames(tellusS.compo.ok)=c(colnames(tellus.grid),
+                              colnames(tellus.compo))
```

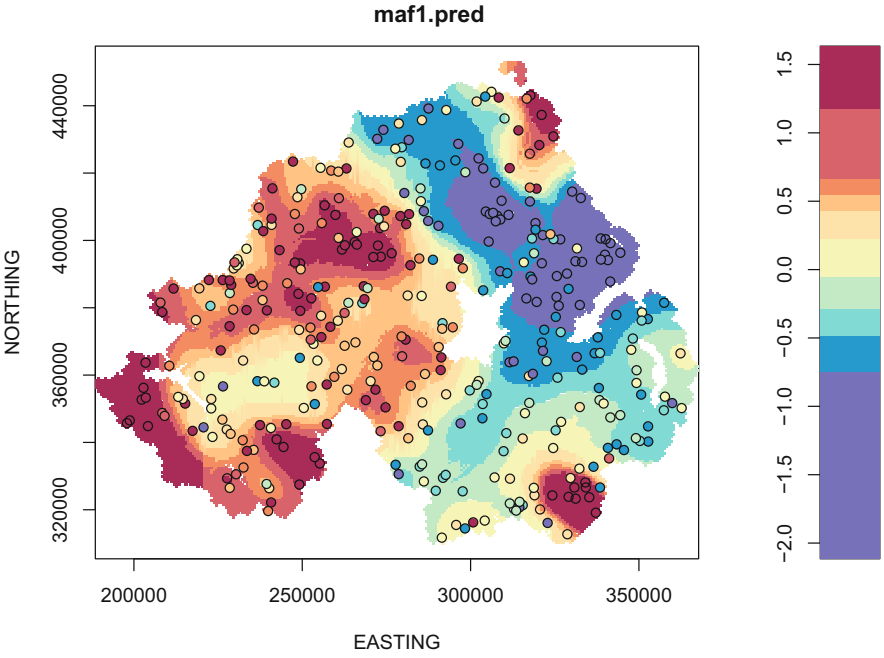

Fig. 6.15 OK estimate of first MAF of the Tellus subcompositon with maf1 sample data superimposed

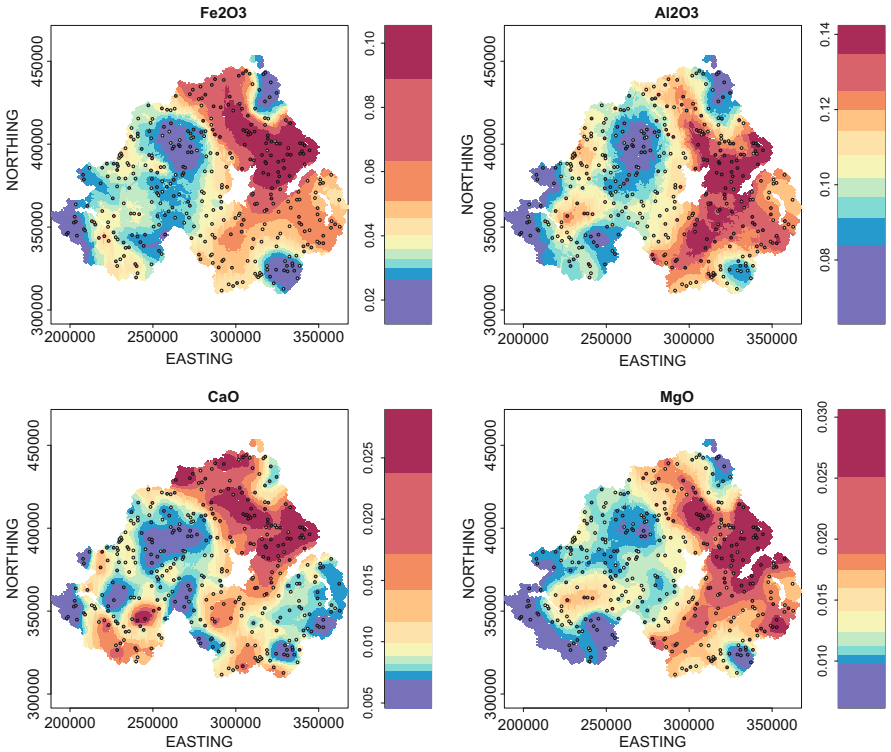

Fig. 6.16 Kriging estimates of Fe_2O_3, Al_2O_3, CaO and MgO based on OK of the MAF factors

Spatial maps in the raw data space are shown in Fig. 6.16.

Histograms of the estimates, true data and associated QQ-plots are shown in Fig. 6.17. Smoothing is evident in the histograms and QQ-plots.

```
> par(mfrow=c(5,3))
> for (i in 1:5){
+ hist(tellusS.compo.ok[,i+2]*100, main="",
+       xlab=names(tellusS.compo.ok)[i+2])
+ hist(tellus.compo[,i], main="true",xlab=names(tellus.compo)[i])
+ x=quantile(tellus.compo[,i],probs = seq(0,1,0.01))
+ y1=quantile(tellusS.compo.ok[tellus.mask,i+2]*100,
+              probs = seq(0,1,0.01),na.rm=T)
+ plot(x,y1,xlab="true",ylab="OK-predicted",asp=1)
+ abline(a=0,b=1)
+ }
```

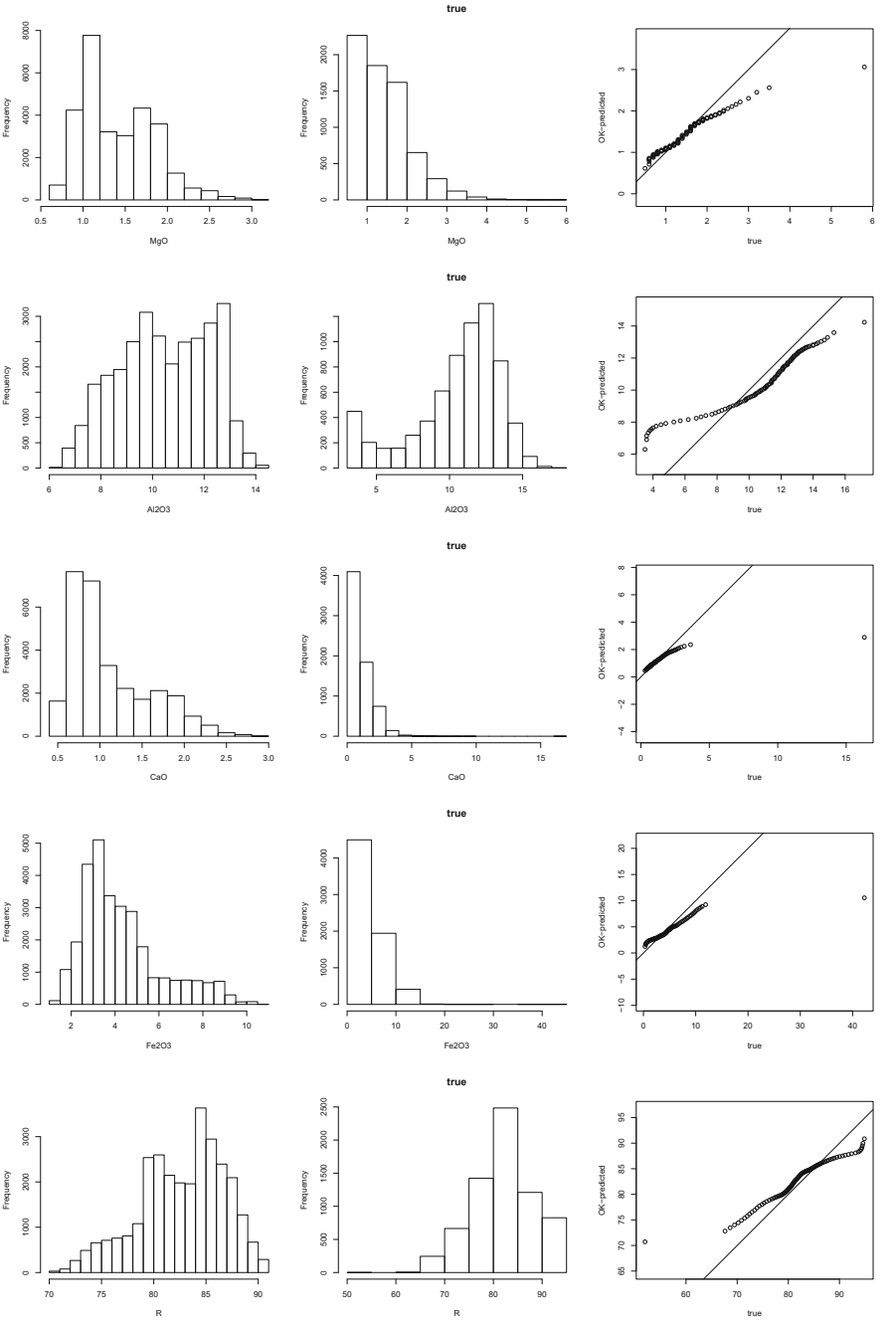

Fig. 6.17 Histograms of estimates, true data and QQ-plots of estimates against true data

Summary statistics also show the smoothing that is a property of every estimation procedure. To see it, though, we need to suppress the report of NA abundance

```
> (tellusS.compo.ok[,-c(1:2)]*100) %>% na.omit %>% summary(dig=3)
```

```
      MgO             Al2O3             CaO
 Min.   :0.615   Min.   : 6.30   Min.   :0.459
 1st Qu.:1.057   1st Qu.: 9.15   1st Qu.:0.769
 Median :1.311   Median :10.46   Median :0.927
 Mean   :1.410   Mean   :10.47   Mean   :1.113
 3rd Qu.:1.747   3rd Qu.:12.01   3rd Qu.:1.405
 Max.   :3.063   Max.   :14.24   Max.   :2.895
     Fe2O3             R
 Min.   : 1.25   Min.   :70.7
 1st Qu.: 2.98   1st Qu.:80.0
 Median : 3.78   Median :83.3
 Mean   : 4.29   Mean   :82.7
 3rd Qu.: 5.03   3rd Qu.:85.8
 Max.   :10.55   Max.   :90.8
```

and compare with the stats of the original data

```
> summary(tellus.compo)
```

```
      MgO             Al2O3             CaO              Fe2O3
 Min.   :0.50   Min.   : 3.5   Min.   : 0.30   Min.   : 0.30
 1st Qu.:0.90   1st Qu.: 9.3   1st Qu.: 0.64   1st Qu.: 2.79
 Median :1.30   Median :11.4   Median : 0.85   Median : 4.19
 Mean   :1.45   Mean   :10.6   Mean   : 1.15   Mean   : 4.65
 3rd Qu.:1.80   3rd Qu.:12.7   3rd Qu.: 1.49   3rd Qu.: 5.69
 Max.   :5.80   Max.   :17.2   Max.   :16.33   Max.   :42.25
       R
 Min.   :52.0
 1st Qu.:78.8
 Median :81.9
 Mean   :82.1
 3rd Qu.:85.8
 Max.   :94.9
```

Problems

6.1 Use of a "unique" Neighbourhood in Cokriging

For the Windarling data in this chapter, compute:

(a) Ordinary cokriging estimates using the unique neighbourhood.
(b) Universal cokriging estimates using the unique neighbourhood.
(c) Compare your results with the OCK and UCK estimates of the composition and the input data.

6.2 Cokriging Invariance with Respect to the Log-Ratio Transformation

(a) For the Windarling data in this chapter, compute ordinary cokriging estimates once using the alr transformation, and once with the ilr transformation. Use a coarse grid, with `Dx = c(5,5)`.
(b) Back-transform the two sets of lr-estimates to compositions, i.e. the first with alr^{-1} and the second with ilr^{-1}.
(c) Compare the output.

6.3 Cokriging of the MAF Factors of the Major Elements of the NGSA Survey in Eastern Australia

(a) Use the isotropic LMC fitted to the experimental direct and cross variograms of the composition of NGSA major elements from `REGION=="EAST"` and `CODE=="Bc"` (bottom soil, coarse fraction) considered in Problems 4.2 and 5.4 to compute OCK estimates of the major elements on a grid of size 25 km by 25 km.
(b) Use the LMRs to variograms of the MAF factors of the composition in Problem 5.4 (a) to derive estimates of the MAF factors on a grid of size 25 km by 25 km, then back-transform first to log-ratios, then to the elemental composition.
(c) Compare the estimates from (a) and (b).

References

Chilès, J. P., & Delfiner, P. (2012). *Geostatistics - Modeling spatial uncertainty* (2 ed., 699 pp.). Hoboken, NJ, USA: Wiley.

Diggle, P. J., & Ribeiro, P. J. J. (2007). *Model based geostatistics* (228 pp.). New York: Springer.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation* (483 pp.). Applied Geostatistics Series. New York, NY, USA: Oxford University Press.

MINES ParisTech/ARMINES (2019). RGeostats: The Geostatistical R Package. Free download from: http://cg.ensmp.fr/rgeostats.

Myers, D. E. (1983). Estimation of linear combinations and co-kriging. *Mathematical Geology, 15*(5), 633–637.

Pawlowsky-Glahn, V., & Olea, R. A. (2004). *Geostatistical analysis of compositional data*. IN DeGraffenreid, Jo Anne. (Ed.), Number 7 in Studies in Mathematical Geology. Oxford University Press.

Schlather, M., Malinowski, A., Oesting, M., Boecker, D., Strokorb, K., Engelke, S., Martini, J., Ballani, F., Moreva, O., Auel, J., Menck, P. J., Gross, S., Ober, U., Ribeiro, P., Ripley, B. D., Singleton, R., Pfaff, B., & R Core Team. (2020). *RandomFields: Simulation and analysis of random fields*. R package version 3.3.8.

Tolosana-Delgado, R. (2006). *Geostatistics for constrained variables: positive data, compositions and probabilities. Application to environmental hazard monitoring* (198 p.). Ph. D. thesis, Universitat de Girona (Spain).

Wackernagel, H. (2003). *Multivariate geostatistics: An introduction with applications* (293 pp.). Berlin: Springer.

Webster, R., & Oliver, M. (2007). *Geostatistics for environmental scientists. Second edition* (271 pp.). Chichester, UK: Wiley.

Chapter 7
Cross-Validation

Abstract Cross-validation is a technique devised to provide a quality assessment of the estimates derived from cokriging and allows appraising different modelling approaches in terms of the choice of variograms and search neighbourhoods.

7.1 Introduction

Cross-validation is a technique devised to provide a quality assessment of the statistical model used for estimation. There are several aspects that are considered: validating the suitability of the variogram model or LMC, deciding between different models of the spatial continuity, as well as decisions about the choice of search neighbourhoods once the variogram model is already chosen. In addition it allows optimising the choice of estimator, e.g. to choose between universal and ordinary cokriging.

In the geostatistical literature there are two main techniques for validation : leave one out (Isaaks & Srivastava, 1989; Goovaerts, 1997; Chilès & Delfiner, 2012; Rossi & Deutsch, 2014) and jackknife validation (Webster & Oliver, 2007). In the case of *leave one out cross-validation*, a single observation is removed from the sample data set at a time and re-estimated based on the neighbouring data. The variogram model tested is the model derived from the entire sample data, as removal of a single observation at a time should have little influence on the variogram estimator and hence the fitted model. In *jackknife* a subset of the sample data is set aside for validation purposes and not used in the inference of the model of spatial continuity. The attribute values of the data within the *jackknife set* are estimated based on the sample data used for the variography and the resultant variogram model. Given that we have introduced univariate and multivariate estimation approaches, both univariate and multivariate cross-validation is considered below.

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_7

7.2 Cross-Validation in the Univariate Case

Irrespective as to what method is applied, at each location within the sample/and or jackknife set a kriging estimate $z_K^*(\mathbf{x}_\alpha)$ will be obtained with associated estimation error variance given by the kriging variance $\sigma_K^2(\mathbf{x}_\alpha)$. For the univariate case, cross-validation can be conducted via `gstat.cv`. The only argument required is the "`gstat`" container providing the data, the information about the variogram model, the chosen estimator and the neighbourhood parameters. Additionally, the function admits an extra argument `nfold` which allows controlling the style of the cross-validation (Bivand et al., 2013). The argument should be given the number of splits to apply to the data for exclusion in the cross-validation. If nfold $>$ 1, the data set is subdivided into nfold subsets and the values for each subset are estimated from the values in the remaining nfold $-$ 1 subsets. If nfold $=$ 1, then leave one out cross-validation is performed. Extreme values are the number of data (the default, providing strict leave one out validation as explained before) and 2 (providing jackknife as explained before), but the function admits any natural number between these two. Even in the case that the "`gstat`" object provided forces the estimates to be obtained via cokriging, the first variable is regarded as the primary variable and estimates for the remaining variables are not computed with full cross-validation information. Thus, this function is only appropriate for validating single variables. For instance, we can get cross-validation results for $\log(\frac{Fe}{R})$, but not for the remaining variables, as follows:

```
> wind.alr1.xvAnis = gstat.cv(wind.alr.ggAnis)

> summary(wind.alr1.xvAnis)

   alr1.pred        alr1.var          observed
 Min.   :0.531   Min.   :0.00212   Min.   :0.111
 1st Qu.:0.660   1st Qu.:0.00216   1st Qu.:0.657
 Median :0.688   Median :0.00217   Median :0.692
 Mean   :0.687   Mean   :0.00219   Mean   :0.687
 3rd Qu.:0.713   3rd Qu.:0.00220   3rd Qu.:0.723
 Max.   :0.909   Max.   :0.00269   Max.   :1.040
    residual           zscore           fold
 Min.   :-0.548   Min.   :-11.16   Min.   :   1
 1st Qu.:-0.024   1st Qu.: -0.52   1st Qu.: 396
 Median : 0.004   Median :  0.08   Median : 792
 Mean   : 0.000   Mean   :  0.00   Mean   : 792
 3rd Qu.: 0.028   3rd Qu.:  0.61   3rd Qu.:1187
 Max.   : 0.359   Max.   :  7.52   Max.   :1582
    Easting          Northing
 Min.   :-235.3   Min.   : 15.5
 1st Qu.:-112.4   1st Qu.: 53.0
 Median : -19.7   Median : 68.0
 Mean   : -15.4   Mean   : 68.7
 3rd Qu.:  84.2   3rd Qu.: 85.8
 Max.   : 204.8   Max.   :122.2

> with(wind.alr1.xvAnis, cor(observed, alr1.pred))
```

```
[1] 0.544
```

```
> with(wind.alr1.xvAnis, cor(zscore, alr1.pred))
```

```
[1] -0.195
```

The descriptive statistics of the output shows a very narrow range for the estimates, and a check of the correlation between estimates and true values suggests moderate correlation between them. Residuals are mostly uncorrelated with predictions, as should be. Note the use here of command `with(data, expression)`, used to evaluate the `expression` given with the variables mentioned extracted from the `data`.

These estimates can be used to compute three diagnostic statistics:

1. The mean error, *ME*, defined as

$$ME = \frac{1}{N}\sum_{\alpha=1}^{N}(z(\mathbf{x}_\alpha) - z_K^*(\mathbf{x}_\alpha))$$

2. The mean squared error, *MSE*, given by

$$MSE = \frac{1}{N}\sum_{\alpha=1}^{N}(z(\mathbf{x}_\alpha) - z_{OK}^*(\mathbf{x}_\alpha))^2$$

3. The mean squared deviation ratio, *MSDR*

$$MSDR = \frac{1}{N}\sum_{\alpha=1}^{N}\frac{(z(\mathbf{x}_\alpha) - z_K^*(\mathbf{x}_\alpha))^2}{\sigma_{OK}^2(\mathbf{x}_\alpha)}$$

The ME should be equal to 0, as kriging is unbiased, the MSE should be small and the $MSDR$ should be close to 1, if the fitted variogram accurately describes the spatial continuity of the attribute (Webster & Oliver, 2007). These error measures can be obtained with the command `xvErrorMeasures` from the companion package to this book

```
> xvErrorMeasures(wind.alr1.xvAnis, output=c("MSDR","MSE"))
```

```
   MSDR      MSE
1.11799 0.00248
```

This function requires an extra argument, `output` accepting one or more of the names of the error measures mentioned: ME, MSE and MSDR.

To make a visual appraisal of the quality of the model, it is useful to also consider the histogram of the errors $\left\{z(\mathbf{x}_\alpha) - z_K^*(\mathbf{x}_\alpha) : \alpha = 1, \ldots, N\right\}$, a scatterplot of the errors against the estimates; and a scatterplot of the estimates against the true values. Ideally the histogram of errors ought to be centred on 0 and have a symmetric shape,

the scatter of the errors against the estimates should show no trends, while the scatter of estimates against true values should ideally be a straight line of slope 1. The latter is hard to achieve and so the most to be realistically expected is that

$$E[Z(\mathbf{x}_0)|Z^*(\mathbf{x}_0)] = Z^*(\mathbf{x}_0)$$

which means that the estimator is *conditionally unbiased*. This may be checked by means of the slope of regression which ought to be close to 1. Schematically:

- The scatterplot between observed values and predictions should have a high positive correlation.
- The distribution of residuals should be symmetric, with a mean of zero and a low variance.
- The distribution of standardised residuals should be symmetric, with a mean of zero and a variance near 1.
- The scatterplot of the residuals versus the predictions should be centred about zero error, satisfying *conditional unbiasedness*. This can also be checked for external trend variables, subdomains or subgroups, to check that no trend is necessary.

These diagrams can be obtained with the following chunk:

```
> par(mfcol=c(2,3))
> # correlation between observed and predicted values
> plot(observed~alr1.pred, main="observed vs. predicted", asp=1,
+      ylab="observed", xlab="predicted", data=wind.alr1.xvAnis)
>   abline(a=0, b=1)
>   abline(lm(observed~alr1.pred, data=wind.alr1.xvAnis), col=2)
> # distribution of residuals
> hist(wind.alr1.xvAnis$residual, xlab="residual",
+      main="residuals histogram")
> qqnorm(wind.alr1.xvAnis$residual,ylab="residual")
>   qqline(wind.alr1.xvAnis$residual, col=2)
> hist(wind.alr1.xvAnis$zscore, xlab="standard error",
+      main = "standard error histogram")
> # uncorrelation of residuals with predictions
> plot(zscore~alr1.pred, data=wind.alr1.xvAnis,
+      xlab="predicted", ylab="standard error",
+      main="standard error vs. predicted")
>   abline(h=0)
> # ... also checked against potential covariables
> boxplot(wind.alr1.xvAnis$residual~windarling$Lithotype,
+         horizontal=T, main="residuals per lithotype",
+         xlab="residual")
```

Results are shown in Fig. 7.1: The residuals and standardised residuals have a reasonably symmetric distribution about 0, but are not normally distributed, because of strong departure from normality in the tails of the distributions. There is no strong difference in distribution by lithotype.

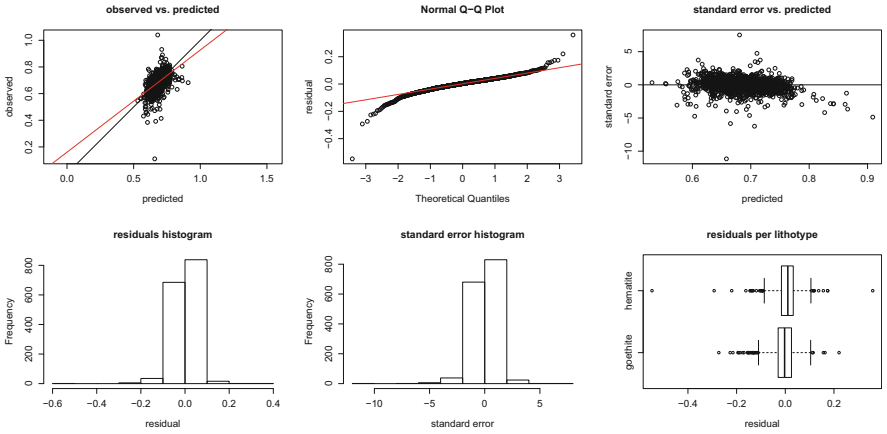

Fig. 7.1 Leave one out cross-validation results of $\log(\frac{Fe}{R})$ based on the LMC `wind.alr.ggAnis`

Finally, one should also check that the residuals are devoid of any further spatial structure. This can be done by mapping them with the methods of Chap. 4 (not shown to save space),

```
> pairsmap(wind.coords, t(t(wind.alr1.xvAnis$residual)),
+           cexrange=c(1,1)*1.5, foregroundcolor=NA)
```

or even better, by checking that the variogram of the residuals is reduced to pure nugget

```
> wind.alr1.xvAnis.vg = gstat(id="residual", formula=residual~1,
+       data=wind.alr1.xvAnis, locations=~Easting+Northing) %>%
+   variogram(cutoff=50, alpha=1:2*90)
> plot(wind.alr1.xvAnis.vg, multipanel=F, col=c("red", "blue"))
```

The output is displayed in Fig. 7.2, showing a quasi perfect removal of spatial structure in both NS and EW directions.

7.3 Multivariate Cross-Validation

In this section, we define a series of tools for multivariate cross-validation. Some of them can be obtained with the standard output provided by `gstat.cv`. For the most interesting results, though, we will need estimates of covariances between residuals, which are beyond the abilities of this function. This will require us to do a manual jackknifing, or use the validation functionalities of "`gmGeostats`".

In this section, instead of re-estimating the value of a single attribute, the entire vector of attributes at the location is estimated. The error measures **ME** and MSE

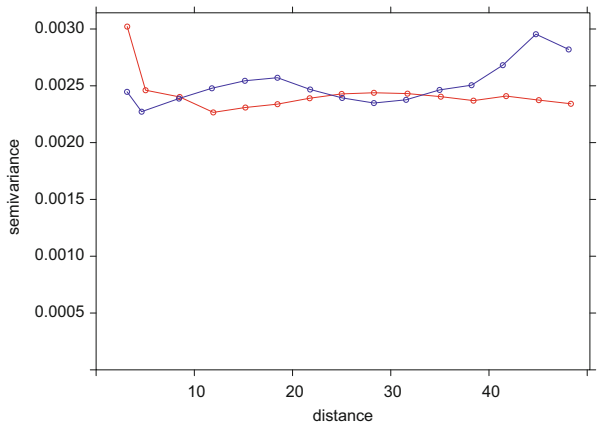

Fig. 7.2 Variogram of the residuals of the leave one out cross-validation of $\log(\frac{Fe}{R})$ based on the LMC `wind.alr.ggAnis`: (red) azimuth=90, (blue) azimuth=180

can then be computed as

$$\mathbf{ME} = \frac{1}{N}\sum_{\alpha=1}^{N}(\mathbf{z}^*_{OCK}(\mathbf{x}_\alpha) - \mathbf{z}(\mathbf{x}_\alpha)); \tag{7.1}$$

$$MSE = \frac{1}{N}\sum_{\alpha=1}^{N}\|\mathbf{z}^*_{OCK}(\mathbf{x}_\alpha) - \mathbf{z}(\mathbf{x}_\alpha)\|^2. \tag{7.2}$$

MSDR can be generalised to the multivariate case in two different ways, shown here for the case of OCK

$$MSDR_1 = \frac{1}{N}\sum_{\alpha=1}^{N}(\mathbf{z}^*_{OCK}(\mathbf{x}_\alpha) - \mathbf{z}(\mathbf{x}_\alpha))^T \boldsymbol{\Sigma}^{-1}_{OCK}(\mathbf{z}^*_{OCK}(\mathbf{x}_\alpha) - \mathbf{z}(\mathbf{x}_\alpha)); \tag{7.3}$$

$$MSDR_2 = \frac{1}{ND}\sum_{\alpha=1}^{N}\sum_{i=1}^{D}\left(\frac{\|\mathbf{z}^*_{\mathbf{i}OCK}(\mathbf{x}_\alpha) - \mathbf{z}_\mathbf{i}(\mathbf{x}_\alpha)\|}{\sigma^{OCK}_{ii}(\mathbf{x}_\alpha)}\right)^2. \tag{7.4}$$

$MSDR_1$ is an affine equivariant quantity and will thus be useful in situations where this invariance from the representation is important, like e.g. when evaluating the cross-validation results from a compositional perspective. $MSDR_2$, on the other hand, depends on the representation itself, and will be useful when the cross-validation results are evaluated in a particularly important representation, for instance in the original compositional units.

To obtain cross-validation estimates for all variables with function `gstat.cv`, we can specify calculation of all residuals (`all.residuals=TRUE`) at the expense of losing information about the estimation error covariance

```
> wind.alr.xvAnis = gstat.cv(wind.alr.ggAnis, remove.all=TRUE,
+                            all.residuals=TRUE, verbose=FALSE)

> summary(wind.alr.xvAnis)

      alr1              alr2              alr3
 Min.   :-0.600   Min.   :-2.195   Min.   :-1.832
 1st Qu.:-0.025   1st Qu.:-0.399   1st Qu.:-0.350
 Median : 0.004   Median :-0.043   Median :-0.050
 Mean   : 0.000   Mean   : 0.012   Mean   : 0.009
 3rd Qu.: 0.029   3rd Qu.: 0.371   3rd Qu.: 0.317
 Max.   : 0.312   Max.   : 2.554   Max.   : 2.507
      alr4              alr5
 Min.   :-2.03    Min.   :-1.437
 1st Qu.:-0.38    1st Qu.:-0.146
 Median :-0.07    Median : 0.017
 Mean   : 0.00    Mean   : 0.001
 3rd Qu.: 0.28    3rd Qu.: 0.157
 Max.   : 3.72    Max.   : 1.176
```

The output produced from the cross-validation is the set of prediction errors generated during the cross-validation. Since the prediction errors are the difference between true values and estimates, the cross-validation estimates can then be derived as the difference between true values and residuals. These results allow calculation of mean errors and mean square absolute errors (Eqs. 7.1–7.2), but none of the MSDR alternatives. The following code and the resulting diagrams of Fig. 7.3 show the comparisons and the calculations:

```
> xv.true = data.frame(alr(wind.compo))
> xv.res = wind.alr.xvAnis
> xv.preds = xv.true-xv.res
> summary(xv.preds)

       Fe               Al2O3              SiO2
 Min.   :0.585   Min.   :-5.17   Min.   :-4.10
 1st Qu.:0.666   1st Qu.:-3.85   1st Qu.:-3.24
 Median :0.688   Median :-3.47   Median :-2.88
 Mean   :0.687   Mean   :-3.44   Mean   :-2.79
 3rd Qu.:0.711   3rd Qu.:-3.04   3rd Qu.:-2.45
 Max.   :0.779   Max.   :-1.63   Max.   : 0.02
       Mn                P
 Min.   :-8.57   Min.   :-6.90
 1st Qu.:-6.35   1st Qu.:-5.77
 Median :-5.81   Median :-5.58
 Mean   :-5.87   Mean   :-5.59
 3rd Qu.:-5.38   3rd Qu.:-5.40
 Max.   :-3.02   Max.   :-4.74

> par(mfcol=c(4,5), mar=c(4,4,3,1))
> for(i in 1:5){
```

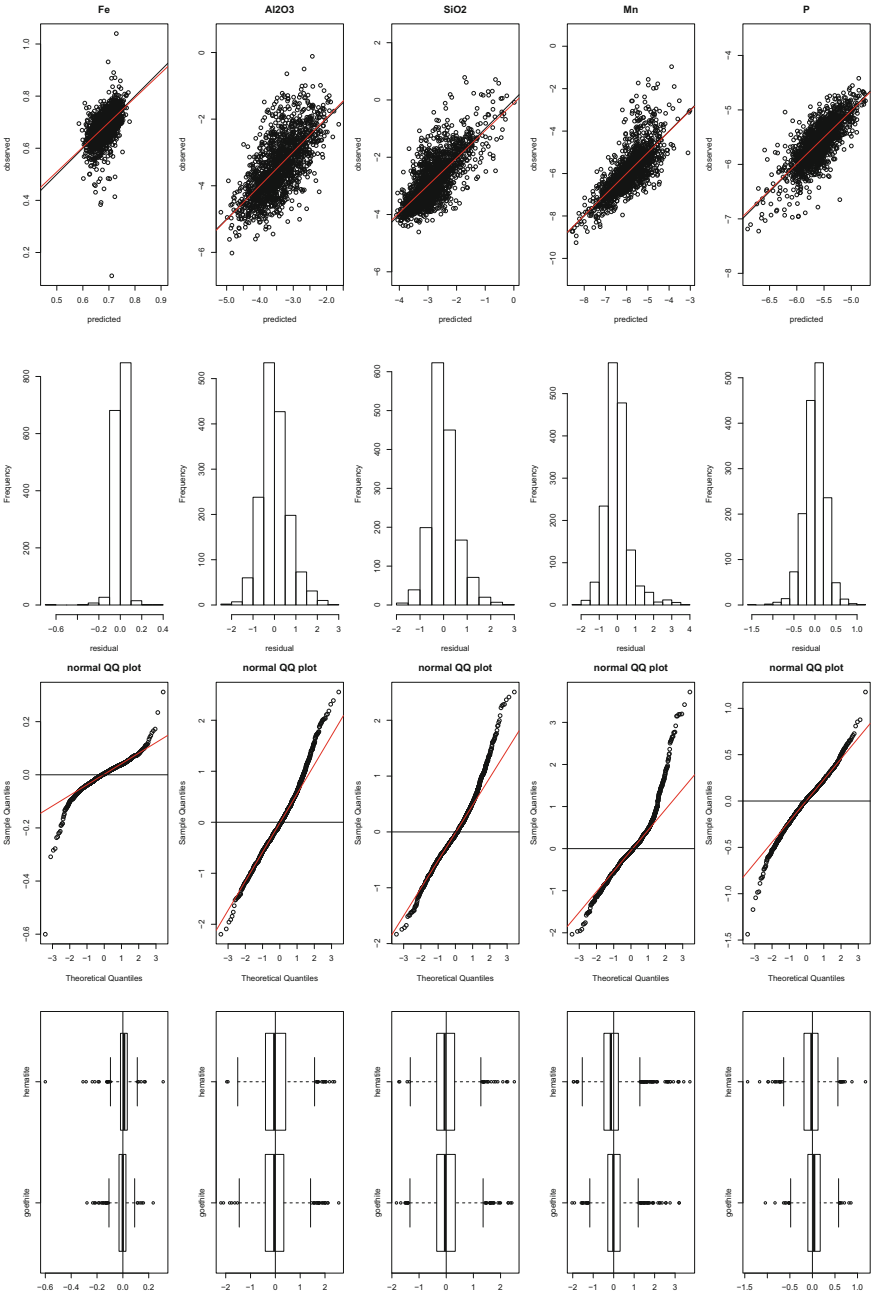

Fig. 7.3 Cross-validation results for the LMC of the alr-transformed Windarling data: Observations against predictions, histograms of residuals, normal QQ-plots for residuals, scatterplots residuals against predictions and boxplots of prediction errors by lithology (Top to bottom)

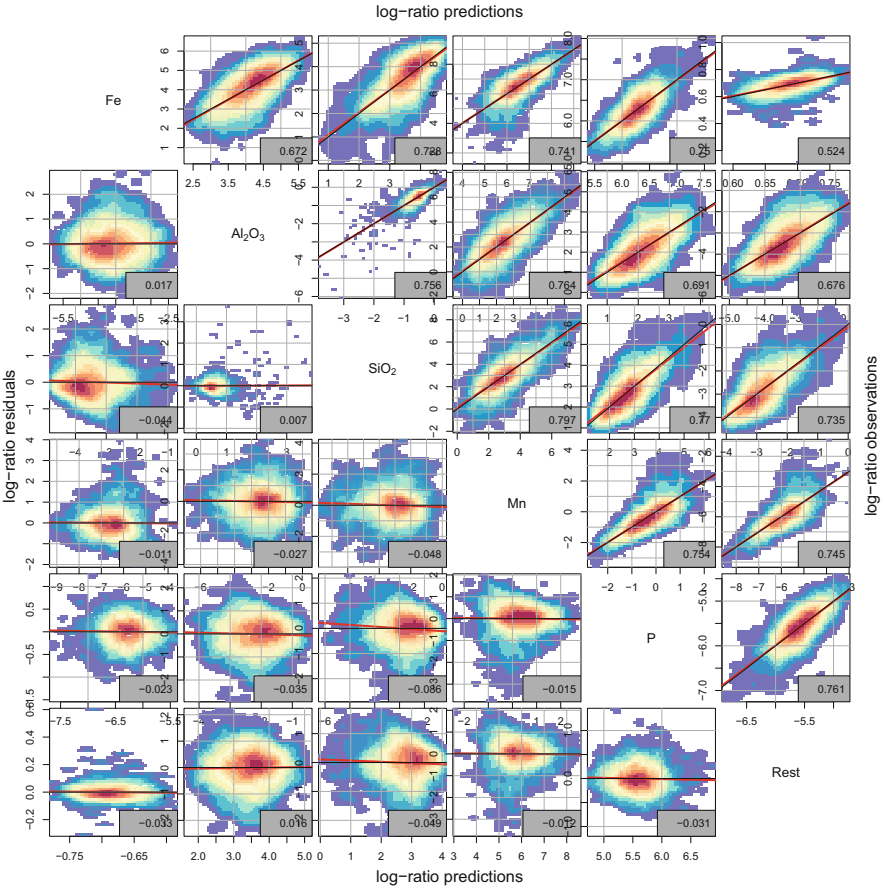

Fig. 7.4 Cross-validation results for the LMC of the alr-transformed Windarling data: log-ratio residuals and log-ratio observations against log-ratio predictions

```
+    plot(xv.true[,i]~xv.preds[,i], ylab="observed",
+         xlab="predicted", asp=1, main=colnames(xv.preds)[i])
+    abline(a=0, b=1)
+    abline(lm(xv.true[,i]~xv.preds[,i]), col=2)
+    hist(xv.res[,i], xlab="residual", main="")
+    qqnorm(xv.res[,i], main="normal QQ plot")
+    qqline(xv.res[,i], col=2)
+    abline(h=0)
+    boxplot(xv.res[,i]~windarling$Lithotype,
+            horizontal=T)
+    abline(v=0)
+ }
```

A plot of the log-ratio residuals against the log-ratio predictions and the log-ratio observations against the log-ratio predictions is shown in Fig. 7.4. These plots

show that the model is adequate for estimation with near-0 correlations between residuals and predictions. Density plots of estimates of the alr-variables against the true values are shown in the 6th column of this density plot matrix. The correlations between cross-validation estimates and true values range from 0.52 to 0.7, which is acceptable.

```
> xv.preds.comp = alrInv(xv.preds, orig=wind.compo)
> xv.true.comp = alrInv(xv.true, orig=wind.compo)
> xv.res.comp = alrInv(xv.res, orig=wind.compo)
> par(mfrow=c(6,6), mar=c(1,1,1,1)/4, oma=c(3,3,3,3))
> for(i in 1:6){
+   for(j in 1:6){
+     if(i==j){
+       plot(c(0,1), c(0,1), bty="n",xaxs="i", yaxs="i",
+            xaxt="n", yaxt="n", pch="")
+       text(0.5, 0.5, labels =
+              expression(Fe, Al[2]*O[3],
+              SiO[2], Mn, P , Rest)[i], cex=1.25)
+     }else if(i>j){
+       # lower tri
+       x = log(xv.preds.comp[,i]/xv.preds.comp[,j])
+       y = log(xv.res.comp[,i]/xv.res.comp[,j])
+       vp.kde2dplot(x,y, add=F, colpalette = spectralcolors)
+       abline(h=0)
+     }else{
+       # upper tri
+       x = log(xv.preds.comp[,i]/xv.preds.comp[,j])
+       y = log(xv.true.comp[,i]/xv.true.comp[,j])
+       vp.kde2dplot(x,y, add=F, colpalette = spectralcolors)
+       abline(a=0, b=1)
+     }
+   }
+ }
> mtext(side = 1:4, line=1, outer=TRUE, text=paste("log-ratio",
+    c("predictions", "residuals","predictions","observations"))
+  )
```

To assess lack of spatial correlation, the direct and cross-variograms of the cross-validation residuals can be computed,

```
> make.gmCompositionalGaussianSpatialModel(data=xv.res.comp,
+   coords = wind.coords, V="alr") %>%
+   variogram(cutoff=50, width=5, alpha=90*1:2) %>%
+   plot
```

The resulting diagram (Fig. 7.5) shows the expected behaviour of flat variograms, around an ideal sill for direct variograms and around 0 for cross-variograms. Finally, residuals could also be represented in maps, for instance with a command such as

```
> pairsmap(data = xv.res.comp, loc=wind.coords, mfrow=c(6,1),
+          cexrange=c(1,1)*1.5, foregroundcolor=NA)
```

as shown in Fig. 7.6, to help assess the lack of dependence of the residuals with the location. If the residuals showed areas of consistently low (under-estimation)

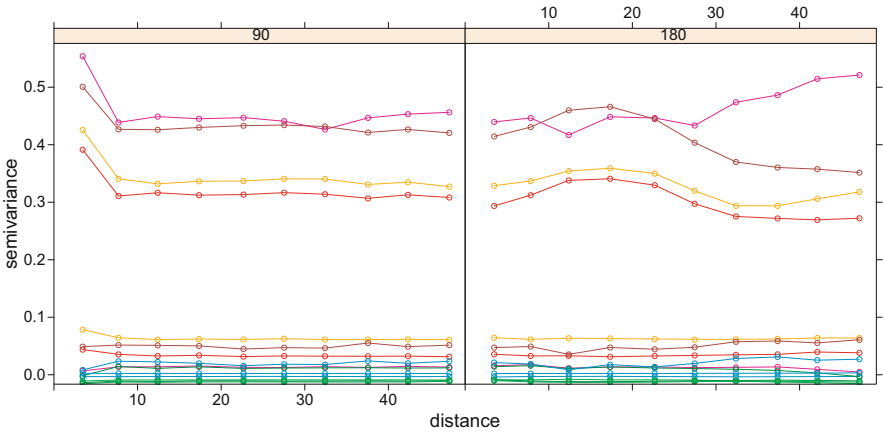

Fig. 7.5 Direct and cross-variograms on two directions (90=EW; 180=NS) of the residuals of a cokriging based leave one out cross-validation of Windarling, model `wind.alr.ggAnis`

or consistently high (over-estimation) values, one could consider the need of incorporating trends into the model.

This is all we can obtain with the standard output of `gstat.cv`. For further calculations involving the full cokriging error structure, one can manually use jackknife, or else use the validation functions of "`gmGeostats`". Here we illustrate how to conduct a manual jackknifing, by means of the Windarling data. The key idea is to split the data into two complementary subsets: `windarlingS` for training and `windarlingJ` for validation (we use a pre-defined subsetting, to ensure reproducibility)

```
> windarlingS = windarling %>%
+   dplyr::filter(Sample.West+Sample.East==1)
> windarling.subset = with(windarling, Sample.West+Sample.East==1)
> windS.coords = wind.coords[windarling.subset,]
> windS.compo = wind.compo[windarling.subset,]
> windarlingJ = windarling %>%
+   dplyr::filter(Sample.West+Sample.East==0)
> windJ.coords = wind.coords[!windarling.subset,]
> windJ.compo = wind.compo[!windarling.subset,]
```

To obtain the jackknife estimates, `windarlingS` will be used to make predictions at the locations of `windarlingJ`, and these will be then compared with their true values. For this we need to clone the "`gstat`" object containing the model to test, but having only the data in the training subset

```
> wind.ng = wind.alr.gg@parameters
> windS.alr.ggAnis = make.gmCompositionalGaussianSpatialModel(
+   data = acomp(windS.compo), coords=windS.coords, V="alr",
+   formula=~1, model = wind.pwlr.lmcAnis, ng=wind.ng)

> windJ.xv=
+   predict(windS.alr.ggAnis, newdata=data.frame(windJ.coords))
```

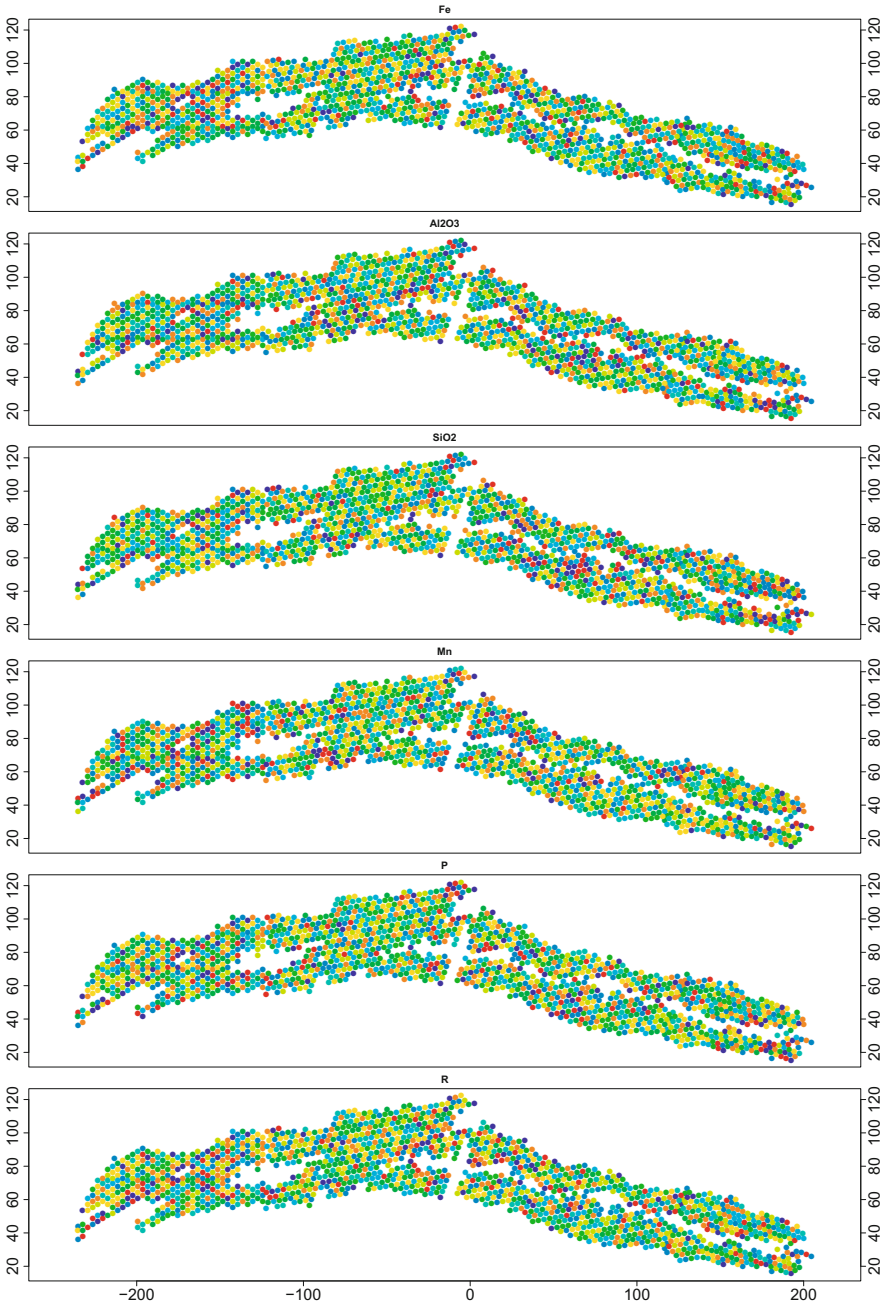

Fig. 7.6 Spatial maps of cross-validation residuals of the Windarling data

```
> colnames(windJ.xv)

 [1] "Easting"       "Northing"      "alr1.pred"
 [4] "alr1.var"      "alr2.pred"     "alr2.var"
 [7] "alr3.pred"     "alr3.var"      "alr4.pred"
[10] "alr4.var"      "alr5.pred"     "alr5.var"
[13] "cov.alr1.alr2" "cov.alr1.alr3" "cov.alr2.alr3"
[16] "cov.alr1.alr4" "cov.alr2.alr4" "cov.alr3.alr4"
[19] "cov.alr1.alr5" "cov.alr2.alr5" "cov.alr3.alr5"
[22] "cov.alr4.alr5"
```

The result has exactly the same shape as a regular cokriging output, as the column names indicate.

Package "gmGeostats" provides this functionality with a pair of commands. First, the user needs to specify the strategy to follow for validating the model. For this step, the package contains a series of functions, depending on the strategy chosen: For leave one out cross-validation, use `LeaveOneOut()`; for an nfold cross-validation, use `NfoldCrossValidation(nfolds, doAll=TRUE)`; and if a simple training/validation split is needed, use `NfoldCrossValidation(nfolds, doAll=FALSE)` and `nfolds` specifying a binary split. The argument `nfolds` can (in both usages) be either an integer specifying the number of splits, or else a factor (or integer vector) specifying which sample is allocated to what split. Once a validation strategy has been chosen, the validation can be obtained by a call to function `validate`; the following code shows how to use it to produce the results we obtained manually:

```
> folds = windarling$Sample.West+windarling$Sample.East+1
> valstgy =  NfoldCrossValidation(nfolds=folds, doAll=FALSE)
> windJ.xv = validate(wind.alr.ggAnis, strategy=valstgy)
```

Figure 7.7 presents marginal diagnostic diagrams analogous to those proposed for univariate cases, including also normalised residuals. This figure is produced with the following chunk:

```
> windJ.xv.true = data.frame(alr(windJ.compo))
> windJ.xv.preds = windJ.xv[,grep("pred", colnames(windJ.xv))]
> windJ.xv.vars = windJ.xv[,grep("var", colnames(windJ.xv))]
> windJ.xv.res = windJ.xv.preds-windJ.xv.true
> colnames(windJ.xv.res)=c("alr1.residuals","alr2.residuals",
+                          "alr2.residuals","alr4.residuals",
+                          "alr5.residuals")
> windJ.xv.zscore=windJ.xv.res/sqrt(windJ.xv.vars)
> par(mfcol=c(5,5), mar=c(4,4,3,1))
> for(i in 1:5){
+   plot(windJ.xv.true[,i]~windJ.xv.preds[,i],ylab="observed",
+        xlab="predicted", asp=1, main=paste("alr",i,sep=""))
+   abline(a=0, b=1)
+   abline(lm(windJ.xv.true[,i]~windJ.xv.preds[,i]), col=2)
+   hist(windJ.xv.res[,i], xlab="residual",main="")
+   qqnorm(windJ.xv.res[,i], main="normal QQ plot")
+   qqline(windJ.xv.res[,i], col=2)
+   plot(windJ.xv.zscore[,i]~windJ.xv.preds[,i],
```

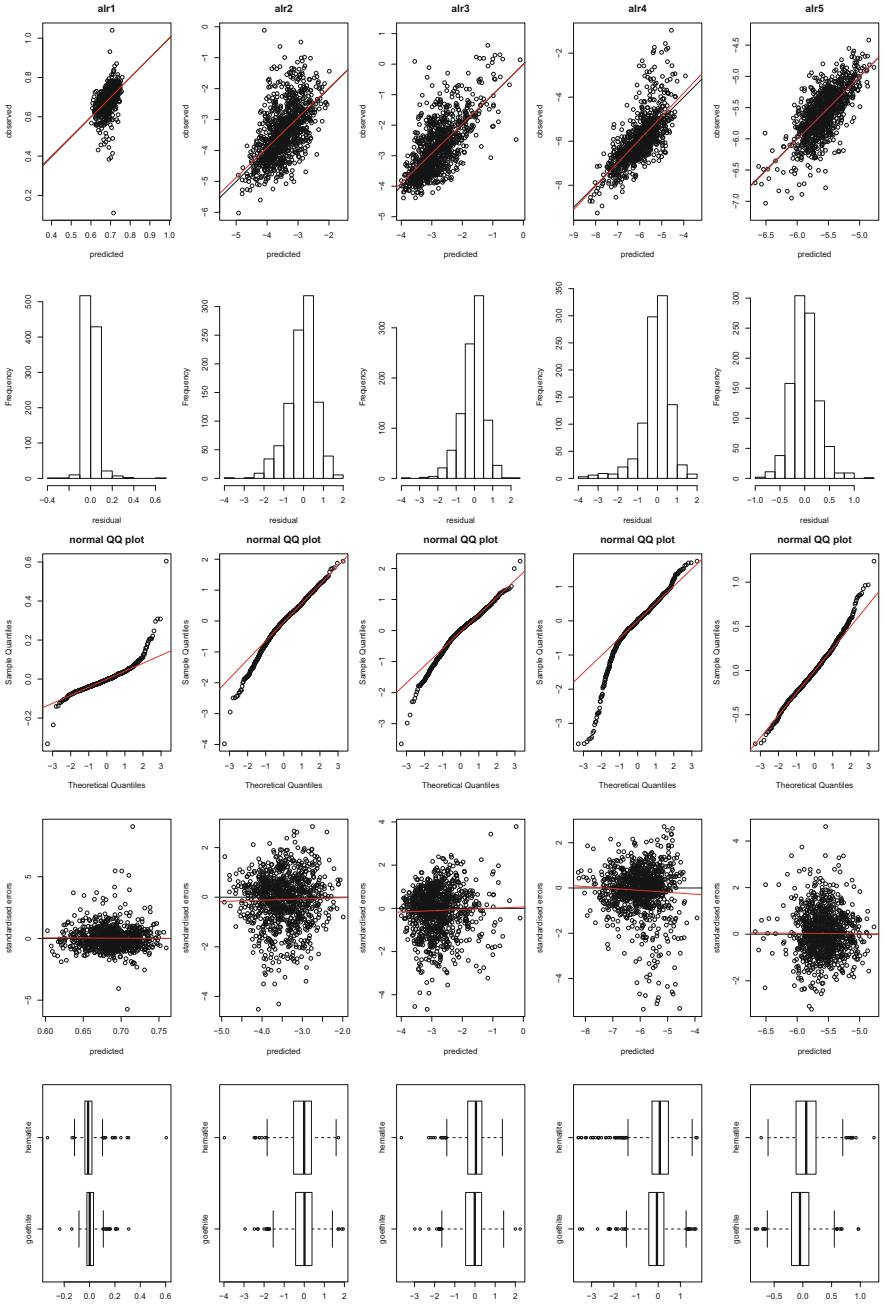

Fig. 7.7 Jackknife validation results for Windarling subset `windarlingJ`, derived from the model based on the entire set and the sample set `windarlingS`, 1st row: true values against estimates; 2nd row: histograms of residuals; 3rd row: QQ-plots of residuals; 4th row: standardised errors against estimates; and 5th row: boxplots of residuals by lithotype

```
+         ylab="standardised errors", xlab="predicted")
+    abline(h=0)
+    abline(lm(windJ.xv.zscore[,i]~windJ.xv.preds[,i]), col=2)
+    boxplot((windJ.xv.res)[,i]~windarlingJ$Lithotype,
+            horizontal=T)
+ }
```

The results show that there is no distinction in the residual profile in relation to lithotype, but a certain departure from normality in the tails of the distributions of the residuals, as evidenced in the QQ-plots.

Estimation error statistics can then be calculated with the function `xvErrorMeasures`

```
> xvErrorMeasures(windJ.xv, observed=windJ.xv.true,
+                 output=c("MSDR1","MSDR2","MSE"))

MSDR1 MSDR2   MSE
 5.33  1.04  1.58
```

The same function provides Mahalanobis norms of the residuals

```
> windJ.mahNorms2 = xvErrorMeasures(windJ.xv,
+                  observed=windJ.xv.true, output="Mahalanobis")
```

These residual norms can be represented in a histogram, or better in a QQ-plot with a χ^2 distribution with 5 degrees of freedom (the number of linearly independent dimensions of the composition) as reference, here reproduced by means of a very large simulated sample of this distribution (Fig. 7.8):

```
> par(mfrow=c(1,2))
> hist(windJ.mahNorms2, breaks=100)
> qqplot(windJ.mahNorms2, rchisq(5000, df=5),
+        xlab="observed quantiles", ylab="chi2(df=5) quantiles")
> abline(a=0, b=1, col=2)
```

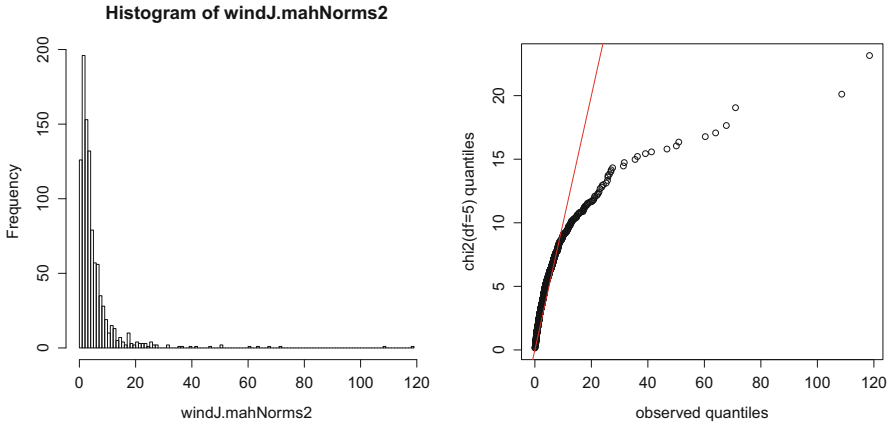

Fig. 7.8 Multivariate cross-validation result for the LMC of the alr-transformed Windarling data: QQ-plot of $MSDR_1$ against the χ^2 distribution with 5 degrees of freedom

7.4 Accuracy and Precision

To further appraise the quality of the statistical model, the accuracy, precision and goodness can be computed. These concepts were introduced by Deutsch (1997) and Rossi and Deutsch (2014), and this section follows the exposition by these authors closely. At each sample location $\mathbf{x}_\alpha$, the kriging estimates and error standard deviations define a normal distribution $F_{\mathbf{x}_\alpha}$ (conditioned on the sample data) with expected value $z_K^*(\mathbf{x}_\alpha)$ and standard deviation $\sigma_K(\mathbf{x}_\alpha)$. By construction there is a probability p such that the true value $z(\mathbf{x}_\alpha)$ falls into the symmetric p-interval bounded by the $\frac{1-p}{2}$ and $\frac{1+p}{2}$ quantiles of this conditional distribution. When all samples are considered, the expected proportion of true values falling into their symmetric p-interval (named the "coverage" of an interval in the statistical literature) should thus be p (or the "confidence" of the interval). One can therefore compare the confidence p with the actual coverage $\hat{\pi}(p)$. To do so define for $p \in [0, 1]$ and $\mathbf{x}_\alpha$ an indicator variable as

$$i(\mathbf{x}_\alpha, p) = \begin{cases} 1 \text{ if } \quad F_{\mathbf{x}_\alpha}^{-1}\left(\frac{1-p}{2}\right) \leq z(\mathbf{x}_\alpha) \leq F_{\mathbf{x}_\alpha}^{-1}\left(\frac{1+p}{2}\right) \\ 0 \text{ otherwise} \end{cases}$$

and put

$$\hat{\pi}(p) = \frac{1}{N} \sum_{\alpha=1}^{N} i(\mathbf{x}_\alpha, p). \tag{7.5}$$

The scatterplot of the calculated proportion $\hat{\pi}(p)$ versus the expected proportion p is called the *accuracy plot*. The *accuracy* A is then defined as

$$A = \int_0^1 a(p)dp,$$

where

$$a(p) = \begin{cases} 1 \text{ if } \hat{\pi}(p) \geq p \\ 0 \quad \text{otherwise} \end{cases}.$$

The *precision* P of the statistical model is defined as

$$P = 1 - 2\int_0^1 a(p)(\hat{\pi}(p) - p)dp. \tag{7.6}$$

The integral in Eq. (7.6) measures the average difference between the confidence p and the coverage $\hat{\pi}(p)$ for those values p at which the model is accurate. The integral of this expression is equal to 0, if $\hat{\pi}(p) = p$ for all p and is positive

otherwise. If $\hat{\pi}(p) = p$ for all p we have an ideal case, giving a precision of 1. The worst case for an accurate model arises when $\hat{\pi}(p) = 1$ for all p. A value of 1 for each value of p essentially means that the distribution at each location is so wide that the true value is contained in every symmetric p-interval. This case corresponds to a value of 0 precision. Another extreme situation occurs if all values of p are inaccurate: In this case the integral of Eq. (7.6) is also 0, and the precision would end up being 1.

A measure that corrects this inappropriate behaviour is the *goodness* G. It takes into account both accurate and inaccurate values as follows:

$$G = 1 - \int_0^1 (3a(p) - 2)(\hat{\pi}(p) - p)dp. \tag{7.7}$$

Here the contributions from values of p at which the model is inaccurate ($\hat{\pi}(p) < p$) are given twice the penalty of the contributions from values of p at which the model is accurate. We have then $G = 1$ if $A = 1$ and $P = 1$, $G = 0.5$ if $A = 1$ and $P = 0$, and for $A = 0$, it follows that $G = 0$. Thus with these additional measures a "good" statistical model has accuracy and goodness statistics close to 1.

Accuracy plots, precision and goodness can be obtained with the functions `accuracy`, `plot.accuracy` (a method of the generic function `plot`) and `precision`:

```
> wind.alr1.xvacc = accuracy(wind.alr1.xvAnis)
> mean(wind.alr1.xvacc)

[1] 0.8

> precision(wind.alr1.xvacc)

precision  goodness
0.911827  0.955793

> plot(wind.alr1.xvacc, main="alr1 accuracy plot")
```

This chunk shows the use of these functions for univariate accuracy and precision. The resulting plot is shown in Fig. 7.9

In the compositional setting, these definitions need to be adapted to reflect the multivariate nature of the data. Otherwise we can only calculate accuracy and precision measures of just one component at a time. This may still be useful for single final variables in the original units, but it does not give any idea of global goodness of the multivariate model. What follows is an extension of the concepts of Deutsch (1997) to the multivariate case.

At each sample location $\mathbf{x}_\alpha$, the cokriging estimates and error covariance matrix define a multivariate normal distribution (conditioned on the sample data) with expected value $\mathbf{z}^*_K(\mathbf{x}_\alpha)$ and cokriging error covariance matrix $\Sigma_K(\mathbf{x}_\alpha)$. Under these conditions, the indicator variable $i(\mathbf{x}_\alpha, p)$ for $p \in [0, 1]$ and $\mathbf{x}_\alpha$ need to be redefined to

$$i(\mathbf{x}_\alpha, p) = \begin{cases} 1 \text{ if } \quad d^2_{AM}\left(\mathbf{z}^*_{OCK}(\mathbf{x}_\alpha), \mathbf{z}(\mathbf{x}_\alpha) | \Sigma_K(\mathbf{x}_\alpha)\right) \leq F^{-1}_{\chi^2(\nu)}(p) \\ 0 \text{ otherwise} \end{cases},$$

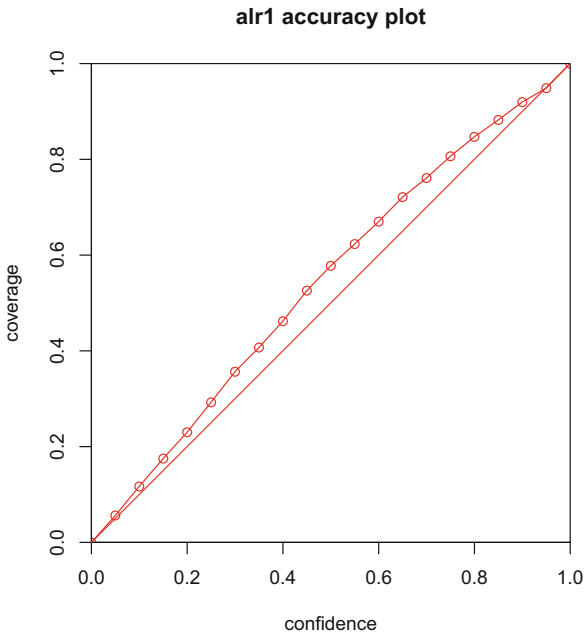

Fig. 7.9 Accuracy plot of the cross-validation of $\log(\frac{Fe}{R})$ based on the LMC `wind.alr.ggAnis`

where $F^{-1}_{\chi^2(\nu)}(p) = r$ is the inverse of the cumulative distribution function of a χ^2-distribution with $\nu = D - 1$ degrees of freedom, and d^2_{AM} is the Aitchison–Mahalanobis distance as introduced in Eq. (3.6). Through this calculation we check whether the true composition is contained within a ball of radius r of the estimated value with respect to the Mahalanobis distance. Accuracy, precision and goodness may then be defined as before, and the interpretation remains the same. The main difference is that here we are working from the assumption of multivariate normality or, within the scope of compositional data, with the assumption of additive logistic normality, as introduced in Chap. 3.

The function `accuracy` and its associated functions can also be used for this calculation, if provided with the output from the jackknife exercise

```
> wind.compo.xvacc= accuracy(windJ.xv, observed=windJ.xv.true,
+                            method = "cokriging" )
> mean(wind.compo.xvacc)

[1] 0.8

> precision(wind.compo.xvacc)

precision  goodness
  0.7949646    0.8896663

> plot(wind.compo.xvacc, main="multivariate accuracy plot")
```

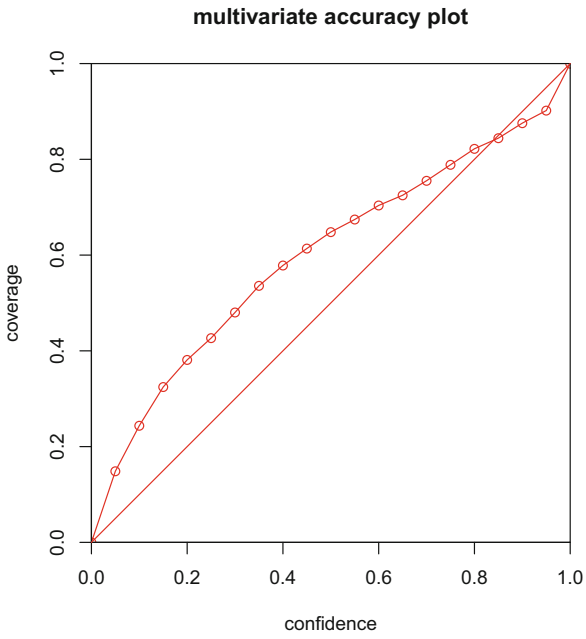

Fig. 7.10 Accuracy plot of the cross-validation of the cokriging model for the Windarling composition

The resulting plot is shown in Fig. 7.10. Note that these are truly compositional measures, because the Mahalanobis distance between cokriged and true value for a composition is independent of the log-ratio representation used for the calculations.

7.5 Some Caveats

The cross-validation approach has been criticised by several authors (Rossi & Deutsch, 2014; Goovaerts, 1997) for the following reasons:

- The same data are used in the cross-validation as were used to infer the variogram model. In an ideal set-up, the use of a jackknife set would arguably be preferable. However, Webster and Oliver (2007) argue that this would amount to a waste of information in real applications.
- The search neighbourhood impacts on the estimates as well, and so the statistics calculated do not simply reflect the variogram model. Any impact the search neighbourhood might have can be avoided through the use of a *unique neighbourhood*. In the case of ordinary kriging, this requires the solution of an $(N + 1) \times (N + 1)$ system of linear equations N times. However the use of a unique neighbourhood is not suitable for large data sets.
- Model parameters such as the relative nugget effect and the behaviour at the origin cannot be cross-validated in most situations.

- The support size for the final model is almost always different from that of the samples and so the quality of the cross-validation might not allow any valid conclusions about the final set of estimates.

Problems

7.1 Cross-Validation of Tellus MAF-OK

(a) Use `gstat.cv` to compute cross-validation statistics of the MAF models for the Tellus subcomposition introduced in Sect. 6.5.
(b) Based on the cross-validation results in (a) calculate accuracy, precision and goodness for each of the four models.

7.2 Multivariate Cross-Validation of Tellus MAF-OK

(a) Use the training–validation subsets strategy to compute cokriging predictions, variances and covariances for the validation subset of the Tellus subcomposition with the MAF models introduced in Sect. 6.5.
(b) Based on the cross-validation results in (a) calculate accuracy, precision and goodness for each of the four models.
(c) Calculate $MSDR_1$.

7.3 Quantitative Kriging Neighbourhood Analysis

We will use the training–validation subsets strategy for the Windarling subcomposition with the MAF models introduced at the end of Sect. 5.5. You can split the data in training and validation by using the logical vector

```
> set = with(windarling0, (Sample.West+Sample.East)==0)
```

(a) Compute cokriging predictions, variances and covariances for the validation subset, using as kriging neighbourhood

```
> wind.ng = KrigingNeighbourhood(nmax=20, nmin=4, maxdist=20)
```

and compute $MSDR_1$ and $MSDR_2$.
(b) Use the code from (a) to define a function `valfun` with a single parameter `nmax`, setting in the call to `KrigingNeighbourhood` the parameter `nmax=nmax`. The function should return a two-component vector with the values for $MSDR_1$ and $MSDR_2$. Check that `valfun(20)` gives the same results as obtained in (a).
(c) Evaluate `valfun` for values of `nmax` between 4 and 50 and plot the results. Can you choose an optimal `nmax` value?

7.4 Cross-Validation of Windarling UCK

(a) Calculate the multivariate cross-validation statistics for UCK of the Windarling composition as discussed in Sect. 6.4.2.
(b) Based on the multivariate cross-validation results in (a) calculate accuracy, precision and goodness for the UCK model.

References

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied spatial data analysis with R, 2nd edition* (405 pp.). New York: Springer Verlag.

Chilès, J. P., & Delfiner, P. (2012). *Geostatistics - Modeling spatial uncertainty* (2 ed., 699 pp.). Hoboken, NJ, USA: Wiley.

Deutsch, C. V. (1997). Direct assessment of local accuracy and precision. In E. Baafi, N. A. Schofield (Eds.), *Geostatistics Wollongong '96 vol. 1* (pp. 102–113). Springer Science & Business Media.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation* (483 pp.). Applied Geostatistics Series. New York, NY, USA: Oxford University Press.

Isaaks, E. H., & Srivastava, R. M. (1989). *An introduction to applied geostatistics* (561 pp.). New York, NY, USA: Oxford University Press.

Rossi, M. E., & Deutsch, C. V. (2014). *Mineral resource estimation* (332 pp.). Dordrecht: Springer.

Webster, R., & Oliver, M. (2007). *Geostatistics for environmental scientists. Second edition* (271 pp.). Chichester, UK: Wiley.

Chapter 8
Multivariate Normal Score Transformation

K. Gerald van den Boogaart, Ute Mueller, and Raimon Tolosana-Delgado

Abstract For the geostatistical simulation of a compositional random function via a Gaussian simulation algorithm, it is often necessary to transform the log-ratio scores of a given composition to multivariate Gaussian variables. In order for the results to be independent of the choice of log-ratio transform, the multivariate normal score transformation needs to be affine equivariant. In this chapter we introduce one such transformation, namely the Gaussian flow anamorphosis and briefly describe its properties.

8.1 Introduction

We have seen that the workflow for the (geo)statistical treatment of compositional data of data is quite straightforward. To summarise it, it is: transform the data with an appropriate log-ratio transformation, analyse and model the transformed scores, and finally back-transform the results obtained (model coefficients, predictions, interpolations, simulations, etc.). Cokriging has been shown to yield interpolations invariant with respect to the choice of log-ratio (Tolosana-Delgado, 2006), a property known as *affine equivariance*.

In the case of geostatistical simulation the invariance property also holds, as long as no further transformation to Gaussianity is necessary. However, for most data sets it is necessary to map data to a Gaussian space prior to simulation. The simplest transform that can be used is a quantile matching with linear interpolation between quantiles and extrapolation to allow extrema by either a power or a hyperbolic function. This transformation is widely used in open source software, such as GSLIB (Deutsch & Journel, 1998). Alternatively the interpolation can be made using a polynomial of sufficiently high order derived from a truncated Hermite series (Rivoirard, 1984; Chilès & Delfiner, 2012). However, while the resulting

Helmholtz Zentrum Dresden Rossendorf, Helmholtz Institute for Resources Technology, Dresden, Germany boogaart@hzdr.de

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*, Use R!, https://doi.org/10.1007/978-3-030-82568-3_8

scores have standard normal marginal distributions, there is no guarantee that they are multivariate normal. This assumption is commonly not checked.

The major drawback of quantile matching is that in the multivariate case variables are transformed individually, and so while marginals are standard normal, there is no guarantee that the multivariate distribution is multivariate normal. Methods currently available for transforming a multivariate data set into a multivariate normal data set are the stepwise conditional transformation (SCT, Leuangthong and Deutsch (2003)), the projection pursuit method (PPM, Barnett et al. (2014)) and the *flow anamorphosis* (FA, Boogaart et al., 2017). Of these methods, only the FA-transformation satisfies the requirement of affine equivariance by construction and as such is best suited to the use in a compositional framework.

8.2 Flow Anamorphosis

The idea of the *FA-transformation* is to deform the underlying space in such a way that the data are shifted towards the centre of a multivariate normal distribution.

Given a set $\{\boldsymbol{\zeta}_i, i = 1, \ldots, n\} \in \mathbb{R}^D$ of vectors with variance–covariance matrix Σ, each vector $\boldsymbol{\zeta}_i$ is assumed to represent the centre of a smoothing kernel K_i. The motion of the kernel can be described by the location of its centre at time t:

$$\boldsymbol{\zeta}_i(t) = (1-t)\boldsymbol{\zeta}_i$$

and the time dependent spread of the smoothing kernel is assumed to be linear in time:

$$\sigma(t) = (\sigma_1 - \sigma_0)t + \sigma_0. \tag{8.1}$$

At time $t = 0$ the centre of the kernel $K_i(0)$ is thus located at $\boldsymbol{\zeta}_i$ and its spread is equal to σ_0 and at time 1, the kernel $K_i(1)$ has centre $\mathbf{0}$ and spread σ_1. Time dependent normal scores are defined relative to the variance–covariance matrix of the data and the spread of the kernel at time t by

$$\mathbf{s}_i(\boldsymbol{\zeta}, t) = \mathbf{L}^{-1}\frac{\boldsymbol{\zeta} - \boldsymbol{\zeta}_i(t)}{\sigma(t)}, \tag{8.2}$$

where $\boldsymbol{\zeta}$ denotes a point of the kernel. The matrix $\mathbf{L}$ is any matrix such that $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$ where $\boldsymbol{\Sigma}$ denotes the variance–covariance matrix of the data. The local movement for the mass of the smoothing kernel K_i is given by:

$$X(\tilde{t}; \boldsymbol{\zeta}, i, t) = \boldsymbol{\zeta}_i(\tilde{t}) + \mathbf{L}\mathbf{s}_i(\boldsymbol{\zeta}, t)\sigma(\tilde{t}) \tag{8.3}$$

and the corresponding speed is

$$\mathbf{v}_i(\boldsymbol{\zeta}, t) = -\boldsymbol{\zeta}_i + \frac{\sigma_1 - \sigma_0}{\sigma(t)}(\boldsymbol{\zeta} - \boldsymbol{\zeta}_i(t)). \tag{8.4}$$

The equation of motion from the raw data to Gaussian space then is given by

$$\frac{\partial}{\partial t}\mathbf{g}(t) = \mathbf{v}(\mathbf{g}(t), t), \tag{8.5}$$

where $\mathbf{g}(t)$ denotes the position of a point at time t and $\mathbf{v}$ is defined as

$$\mathbf{v}(\boldsymbol{\zeta}, t) = \frac{\sum_i w_i(\boldsymbol{\zeta}, t)\mathbf{v}_i(\boldsymbol{\zeta}, t)}{\sum_i w_i(\boldsymbol{\zeta}, t)} \tag{8.6}$$

with weights at time t defined by

$$w_i(\boldsymbol{\zeta}, t) = \exp\left(-\frac{1}{2}\|\mathbf{s}_i(\boldsymbol{\zeta}, t)\|^2\right),\ i = 1, \ldots, n. \tag{8.7}$$

The back-transform from Gaussian space back to raw data space is achieved via a reversal of the flow :

$$\frac{\partial}{\partial t}\mathbf{r}(t) = -\mathbf{v}(\mathbf{r}(t), 1 - t).$$

Both equations of motion are solved numerically.

8.3 Properties

The above flow transformation has the following properties (Boogaart et al., 2017; Mueller et al., 2017):

- The flow is invariant under linear transformations.
  If the points and smoothing kernel centres are subjected to the linear transformation

  $$\boldsymbol{\eta} = \mathbf{A}\boldsymbol{\zeta}$$

  and the motion of points at time t is given by $\boldsymbol{\eta}(t) = \mathbf{A}\boldsymbol{\zeta}(t)$, then the motion field at the transformed locations is

  $$\mathbf{v}_i(\boldsymbol{\eta}, t) = \mathbf{A}\mathbf{v}_i(\boldsymbol{\zeta}, t). \tag{8.8}$$

- If the points and smoothing kernel centres are subjected to the affine transformation

$$\boldsymbol{\eta} = \mathbf{A}\boldsymbol{\zeta} + \mathbf{b}$$

and the motion of points at time t is given by $\boldsymbol{\eta}(t) = \mathbf{A}\boldsymbol{\zeta}(t) + (1-t)\mathbf{b}$, then the speed of the ith kernel is given by $\mathbf{v}_i(\boldsymbol{\eta}, t) = \mathbf{A}\mathbf{v}_i(\boldsymbol{\zeta}, t) - \mathbf{b}$.
- Since the scaling impacts on the time dependent normal scores resulting in slightly different speeds of the flow field, the normal scores depend on the initial spread of the kernels σ_0 and on σ_1. The parameter σ_0 controls the strength of deformation while σ_1 controls the ranges of the transformed variables (Boogaart et al., 2017).
- Neighbourhood relationships are preserved: Points which are originally proximal are transformed to points in normal score space that are proximal.
- The transformed data are decorrelated by construction and moreover, often also spatially decorrelated.

8.4 Application to Windarling Data

In Chap. 3 we saw that the Windarling data are not ALN-normal. This is not corrected just by transforming each marginal alr variable to normality, for instance with quantile matching (Fig. 8.1). This is also confirmed via multivariate normality tests, such as the Mardia, Henze-Zirkler or energy tests in the package "MVN" (Korkmaz et al., 2014). Therefore a truly multivariate normality transformation is needed.

```
         Test     H p value MVN
royston     1     0       1   1
hz          1 4.07        0   1
energy      1 18.9        0   1
```

To transform the data to multivariate normality, the flow anamorphosis can be applied. This is available with the `Anamorphosis` code in "`gmGeostats`". As a first step the data need to be loaded and transformed to alr or ilr-scores. Since the flow anamorphosis is affine equivariant, it does not matter what transformation is used. Here the ilr transformation is used.

```
> wind.ilr= ilr(wind.compo,V=ilrBase(wind.compo))
```

The deformation parameter σ_0 is set to 0.2, and the range parameter σ_1 to 1.2.

```
> library("gmGeostats")

> fana.wind = ana(wind.ilr, sigma0=0.2 , sigma1=1.2)
> wind.ns = fana.wind(wind.ilr)

> plot(data.frame(wind.ns), panel=vp.kde2dplot,
+      colpalette = spectralcolors)
```

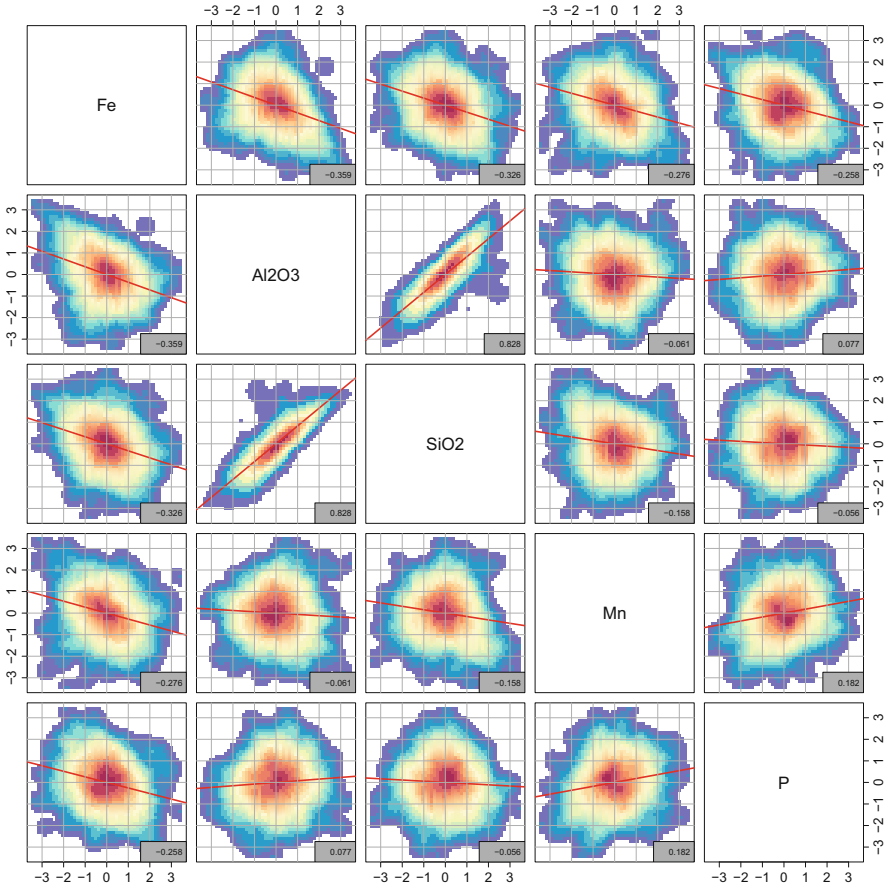

Fig. 8.1 Bivariate density plots of normal scores for Windarling data derived from quantile matching

The scatterplots of the output data from the flow anamorphosis suggest little correlation between the normal scores (Fig. 8.2). A graphical check of normality (Fig. 8.3) further confirms that the level of deformation applied to the raw data suffices to transform the data to normal scores. The histograms for the transformed data have the shape of histograms for normally distributed data and the normal QQ-plots only indicate departures from the normal distributions in the tails, and these departures are slight.

```
> par(mfcol=c(2,5), mar=c(3,3,2,1))
> for(i in 1:5){
+   hist(wind.ns[,i], main=colnames(wind.ns)[i])
+   qqnorm(wind.ns[,i], main="")
+   qqline(wind.ns[,i], col=2)
+ }
```

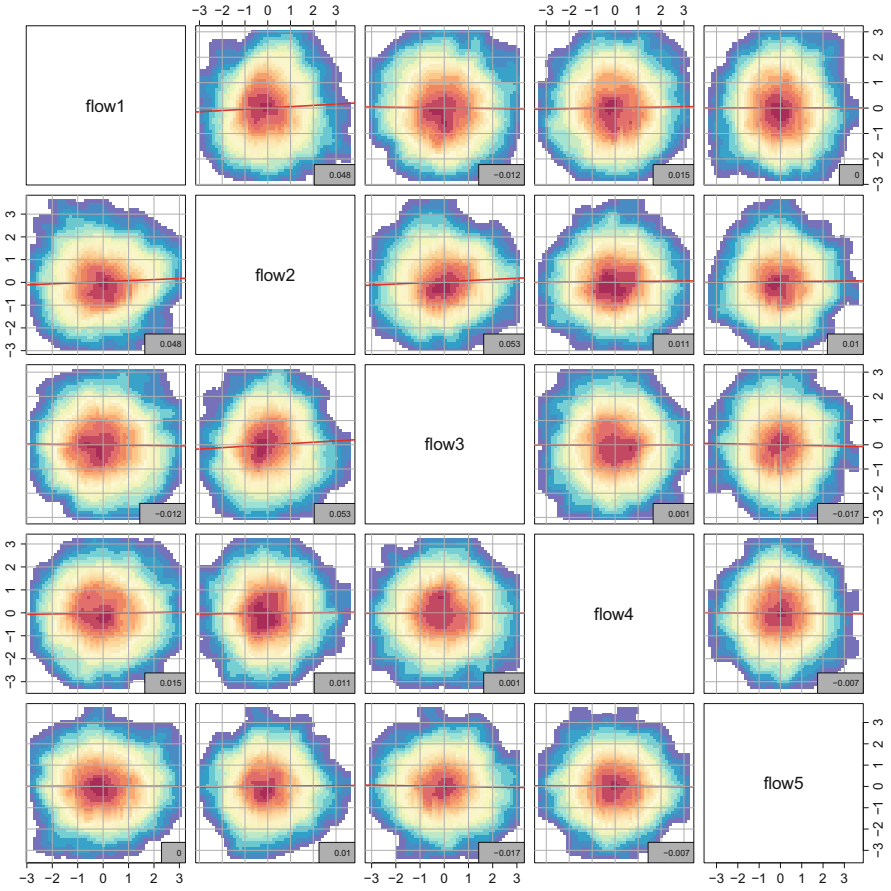

Fig. 8.2 Bivariate density plots of normal scores for Windarling data derived from FA

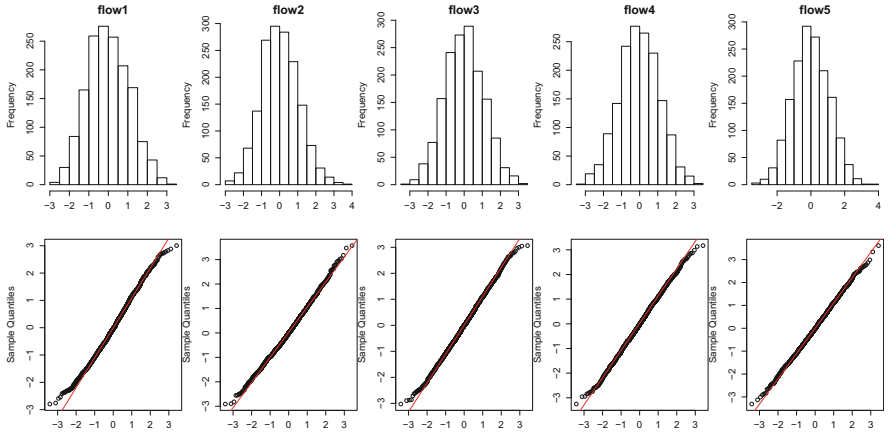

Fig. 8.3 Normality check for normal scores of Windarling data derived from FA

However, as confirmed by various multivariate normality tests, the transformed data fail to be multivariate normal, and two of the five marginals fail the univariate normality test. These are $flow1$ and $flow2$ and both show departure from the bisector in the tails.

```
> library(MVN)
> mvn(wind.ns, mvnTest="mardia", univariateTest="SW", desc=F)

$multivariateNormality
             Test         Statistic              p value
1 Mardia Skewness  117.244044433137 8.33489954583645e-11
2 Mardia Kurtosis -7.44157733830992  9.9475983006414e-14
3             MVN              <NA>                  <NA>
  Result
1     NO
2     NO
3     NO

$univariateNormality
          Test  Variable Statistic   p value Normality
1 Shapiro-Wilk   flow1       0.996    0.0001    NO
2 Shapiro-Wilk   flow2       0.997    0.0121    NO
3 Shapiro-Wilk   flow3       0.998    0.1502    YES
4 Shapiro-Wilk   flow4       0.999    0.2952    YES
5 Shapiro-Wilk   flow5       0.999    0.7269    YES

> mvn(wind.ns, mvnTest="hz", univariateTest="Lillie", desc=F)

$multivariateNormality
           Test    HZ p value MVN
1 Henze-Zirkler 0.919   0.803 YES

$univariateNormality
                            Test  Variable Statistic
1 Lilliefors (Kolmogorov-Smirnov)   flow1      0.0281
2 Lilliefors (Kolmogorov-Smirnov)   flow2      0.0271
3 Lilliefors (Kolmogorov-Smirnov)   flow3      0.0148
4 Lilliefors (Kolmogorov-Smirnov)   flow4      0.0183
5 Lilliefors (Kolmogorov-Smirnov)   flow5      0.0153
    p value Normality
1    0.0054    NO
2    0.0088    NO
3    0.5458    YES
4    0.2238    YES
5    0.4963    YES

> mvn(wind.ns, mvnTest="energy", univariateTest="AD")

$multivariateNormality
         Test Statistic p value MVN
1 E-statistic      2.17       0  NO

$univariateNormality
              Test  Variable Statistic   p value Normality
1 Anderson-Darling   flow1       1.453    0.0009    NO
```

```
2 Anderson-Darling   flow2       1.043    0.0096    NO
3 Anderson-Darling   flow3       0.318    0.5361    YES
4 Anderson-Darling   flow4       0.331    0.5134    YES
5 Anderson-Darling   flow5       0.288    0.6185    YES

$Descriptives
         n     Mean Std.Dev   Median   Min  Max   25th  75th
flow1 1582  0.00657    1.06 -0.03868 -2.79 3.01 -0.743 0.733
flow2 1582  0.03046    1.02 -0.00481 -2.91 3.57 -0.666 0.719
flow3 1582 -0.00501    1.07 -0.00503 -3.02 3.07 -0.742 0.756
flow4 1582 -0.01674    1.09 -0.02363 -3.25 3.17 -0.720 0.770
flow5 1582  0.00549    1.07 -0.02683 -3.31 3.61 -0.742 0.749
         Skew Kurtosis
flow1  0.1371   -0.409
flow2  0.1857   -0.034
flow3  0.0590   -0.233
flow4 -0.0117   -0.232
flow5  0.0465   -0.171
```

Spatial maps of the normal scores are shown in Fig. 8.4

```
> pairsmap(data = wind.ns, loc=wind.coords, cexrange=c(1,3)*0.5,
+          mfrow=c(5,1), foregroundcolor=NA)
```

To check the spatial decorrelation of the FA-transformed data, empirical direct and cross variograms are calculated and plotted (Fig. 8.5)

```
> wind.ns.gg = make.gmMultivariateGaussianSpatialModel(
+   data = data.frame(wind.ns), coords = wind.coords )

> wind.ns.vg = variogram(wind.ns.gg, cutoff=40, width=3.5)
> variogramModelPlot(wind.ns.vg, commonAxis=TRUE)
```

Note the use of the function `make.gmMultivariateGaussianSpatial Model` to create a spatial model object for non-compositional variables. The experimental cross variograms are quite flat and the values are close to 0, similar to what was observed in the case of the variograms and cross variograms of the MAF factors of the raw data in Chap. 4. There appears to be some remnant spatial correlation between the variables flow1 and flow5, and between flow2, flow3 and flow4. The spatial correlation of these variables with flow1 and flow5 is negligible.

To further assess the spatial decorrelation, the numerical measures introduced in Chap. 4 will be computed. The plots further support the observation that the FA normal scores are not sufficiently spatially decorrelated, as shown by high values (compared to 0) in particular for the first eight lags in both the relative and absolute deviations from diagonality (Fig. 8.6).

```
> par(mfrow=c(1,2))
> aux1=spatialDecorrelation(wind.ns.vg,method="rdd")
> mean(aux1$gamma)

[1] 0.265

> plot(gamma~dist,aux1, type="o",ylab="rdd",main="rdd")
> aux2=spatialDecorrelation(wind.ns.vg,method="add")
> mean(aux2$gamma)
```

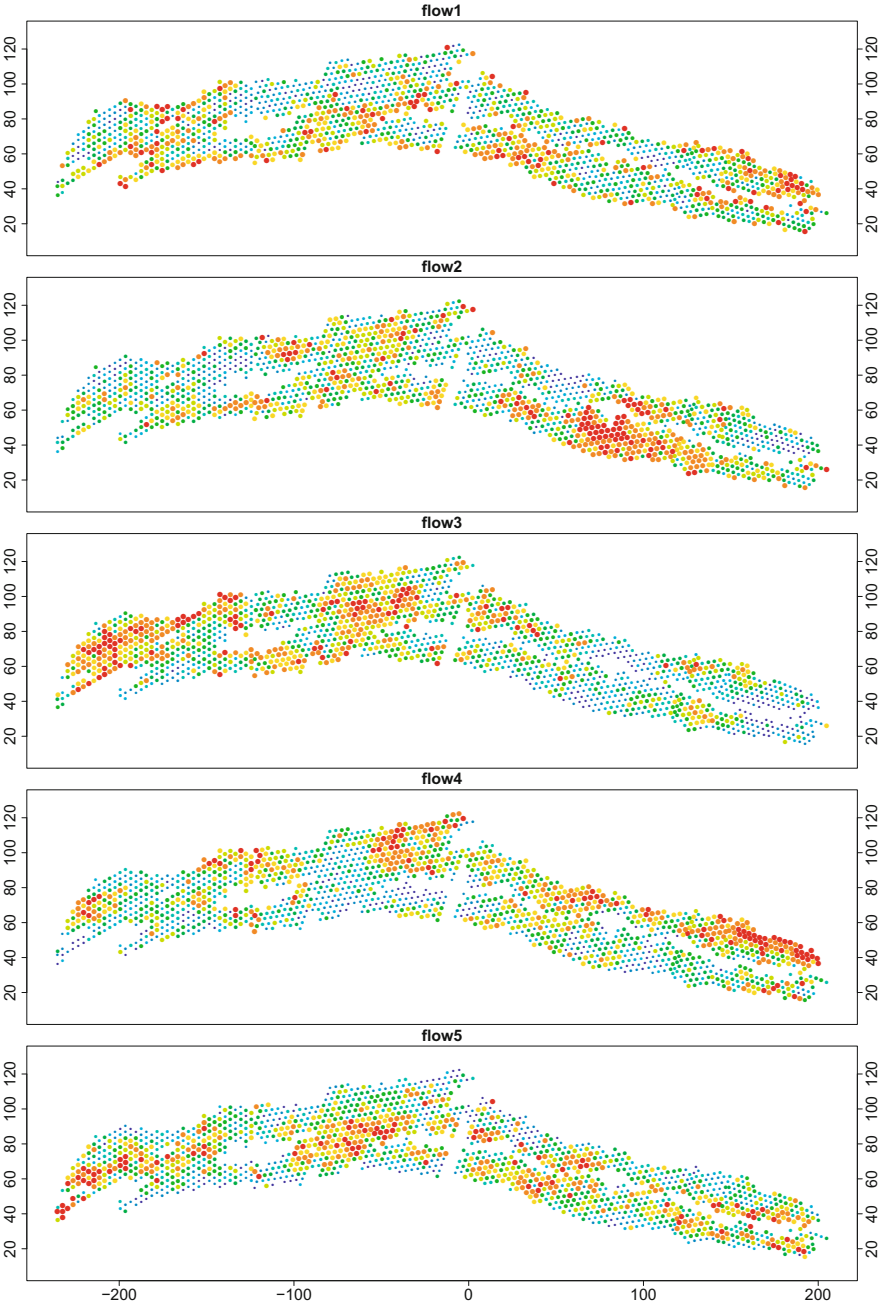

Fig. 8.4 Spatial maps of normal scores for Windarling data derived from FA

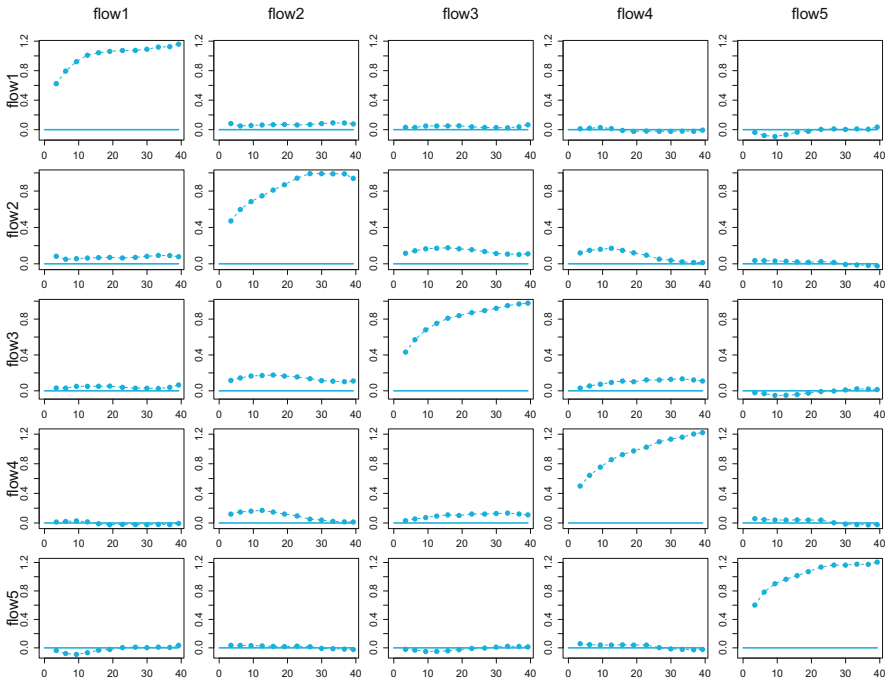

Fig. 8.5 Direct and cross variograms of normal scores for Windarling data derived from FA

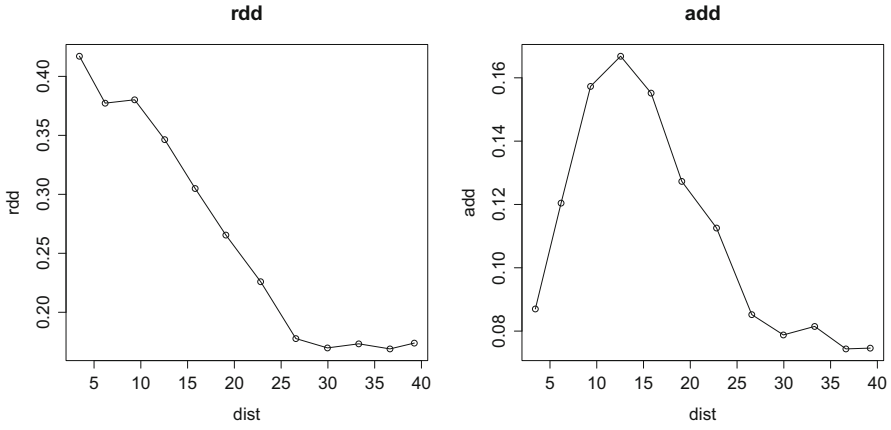

Fig. 8.6 Spatial decorrelation measures for FA normal scores of Windarling data

```
[1] 0.11

> plot(gamma~dist,aux2, type="o",ylab="add",main="add")
> par(mfrow=c(1,1))
```

We can therefore conclude that the FA-transformed data cannot be deemed to be spatially decorrelated in this case and that some further processing will be required before using the data for simulation.

Problems

8.1 Impact of the Tuning Parameters on the Flow Anamorphosis
In this set of exercises, we will study the impact of the parameters σ_0 and σ_1 on the results of the FA-transformation. As in this chapter, we will apply the anamorphosis to the Windarling set.

(a) Apply FA to wind.ilr for $\sigma_0 = 0.5, 0.25, 0.1, 0.05, 0.01$ and $\sigma_1 = 1$. Comment on the multivariate normality of the output. Which of the values for σ_0 provides the best compromise between space deformation and normality?
(b) For the value of σ_0 in (a) that provided the most suitable results, determine the effect of the following settings of σ_1 on the output: $\sigma_1 = 1, 1 + 0.5\sigma_0, 1 + \sigma_0, 1 + 2\sigma_0$.

8.2 Iterated Anamorphosis
Determining the appropriate space deformation can be time consuming and so an iterative approach might be preferable. This approach involves choosing a fixed pair of values for σ_0 and σ_1, and applying the anamorphosis repeatedly until no further change in the scores is observed.

(a) Apply FA to wind.ilr for $\sigma_0 = 0.2$ and $\sigma_1 = 1.2$. Ensure that the input data are sphered. Call the output set $FA^{(1)}$.
(b) Now calculate a sequence of FA-transformed scores by applying the flow anamorphosis with $\sigma_0 = 0.2$ and $\sigma_1 = 1.2$ but without spherification to $FA^{(n-1)}$, $n = 2, 3, \ldots$. After each iteration determine the ranges of the output variables and check their multivariate normality via the package "MVN". Can you achieve multivariate normality and if so, how many iterations are required?

8.3 FA-Transformation of a Subcomposition Within the Tellus Subset
Determine the most suitable parameters to transform the Tellus subcomposition {MgO, Fe_2O_3, CaO, Al_2O_3, Rest} within the subset (identified by Flag=1) to multivariate normality.

8.4 FA-Transformation of the Major Elements NGSA Data Set to Normal Scores

(a) Consider the NGSA data set, filtering it to keep only those observations from REGION=="EAST" and CODE=="Bc" (bottom soil, coarse fraction). Select

the major elements of the NGSA data set, those measured with XRF (thus having that string on the column name). Replace the values below detection limit by the detection limit itself. Close the data through addition of a rest variable and transform the composition to ilr-space. Now apply FA to the ilr data for $\sigma_0 = 0.2, 0.1, 0.05, 0.01$ and $\sigma_1 = 1 + \sigma_0$.

(b) For each of the FA transformed data sets thus obtained, test for multivariate normality and hence decide what value of σ_0 to apply.

References

Barnett, R. M., Manchuk, J. G., & Deutsch, C. V. (2014). Projection pursuit multivariate transform. *Mathematical Geosciences, 46*(2), 337–360.

Boogaart, K. G. v. d., Mueller, U., & Tolosana-Delgado, R. (2017). An affine equivariant multivariate normal score transform for compositional data. *Mathematical Geosciences, 49*(2), 231–251.

Chilès, J. P., & Delfiner, P. (2012). *Geostatistics - Modeling spatial uncertainty* (2 ed., 699 pp.). Hoboken, NJ, USA: Wiley.

Deutsch, C., & Journel, A. (1998). *GSLIB - Geostatistical software library and user's guide* (2nd ed., 368 p., and one compact disk). New York, NY, USA: Oxford University Press.

Korkmaz, S., Goksuluk, D., & Zararsiz, G. (2014). Mvn: An r package for assessing multivariate normality. *The R Journal, 6*(2), 151–162.

Leuangthong, O., & Deutsch, C. V. (2003). Stepwise conditional transformation for simulation of multiple variables. *Mathematical Geology, 35*(2), 155–173.

Mueller, U., Boogaart, K. G. v. d., & Tolosana-Delgado, R. (2017). A truly multivariate normal score transform based on Lagrangian flow. In *Geostatistics Valencia 2016* (pp. 107–118). Cham: Springer.

Rivoirard, J. (1984). Une methode d'estimation du recuperable local multivariable. *Note 894, CGMM, Mines-Paris Tech*, 10 pp.

Tolosana-Delgado, R. (2006). *Geostatistics for constrained variables: positive data, compositions and probabilities. Application to environmental hazard monitoring* (198 p.). Ph. D. thesis, Universitat de Girona (Spain).

Chapter 9
Simulation

Abstract In this chapter we look at the geostatistical simulation of compositional data. The workflow typically consists of first applying a log-ratio transformation, then testing the transformed data for multivariate normality, applying an affine equivariant multivariate normal score transformation, if necessary, modelling the spatial continuity of the normal scores, followed by simulation and back-transformation. Several algorithms are available for simulating Gaussian random functions. These include LU decomposition simulation, sequential Gaussian simulation and turning bands simulation, which will be introduced briefly.

9.1 Introduction

Geostatistical simulation is used to obtain multiple equiprobable realisations of the random function across the study region. These realisations can be used to make uncertainty assessments, but also to change support from point support to larger volumes.

A key assumption is that the variable(s) under consideration are multivariate multi-Gaussian, so that their distribution can be characterised by the mean and covariance alone. If the condition of multivariate multi-Gaussianity is not satisfied, the variables under study need to be transformed to Gaussian variables prior to simulation, and the Gaussian random function obtained in that manner is then simulated. The model describing the spatial continuity is based on the transformed data if a transformation needed to Gaussian variables was necessary, otherwise the raw data are used in the variography. Following the simulation the results are back-transformed to the raw data space, if required, prior to any evaluation or other postprocessing.

Several simulation algorithms are available in **R**, in particular in the package "`RandomFields`". What follows is a brief report of those traditionally used in the geostatistical literature, or of special interest for multivariate Gaussian random fields. Some of them are available in packages "`gstat`" or "`gmGeostats`", as explained below. Complete accounts of geostatistical simulation techniques can be found in the works of Lantuéjoul (2002) and Chilès and Delfiner (2012).

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_9

9.2 Simulation Algorithms

9.2.1 Sequential Gaussian Simulation

One of the most popular algorithms is *sequential Gaussian simulation* (*SGS*, Journel and Huijbregts (1978); Goovaerts (1997); Chilès and Delfiner (2012)) because of the simplicity of the implementation, irrespective as to whether the univariate or multivariate case is considered.

Sequential Gaussian simulation is based on the simple (co-)kriging of the data. The procedure for generating a single realisation on a simulation grid is as follows:

1. Define a random path visiting all grid nodes exactly once.
2. At grid node $\mathbf{x}_0$ compute the simple kriging mean $\boldsymbol{\mu}_{SK}(\mathbf{x}_0)$ and simple kriging error covariance matrix $\boldsymbol{\Sigma}_{SK}(\mathbf{x}_0)$ based on the sample observations and all previously simulated vectors of attributes.
3. Draw a random observation $\mathbf{z}(\mathbf{x}_0)$ from $\mathcal{N}(\boldsymbol{\mu}_{SK}(\mathbf{x}_0), \boldsymbol{\Sigma}_{SK}(\mathbf{x}_0))$ and add it to the list of already simulated vectors.
4. Move to the next location along the path and repeat the previous two steps.
5. Repeat until all locations along the path have been populated with simulated attribute vectors.

To generate a further realisation, the process outlined above is repeated.

SGS is the standard algorithm available in "`gstat`", activated when using function `predict.gstat` with an extra argument `nsim` other than `NULL`. In "`gmGeostats`", you can specify an SGS simulation design by means of the function `SequentialSimulation`, which can then be fed as an extra argument `pars` to the `predict.gmSpatialModel`,

```
> sgs = SequentialSimulation(nsim, ng, ...)
> predict(model, newdata, pars=sgs)
```

Note that a kriging neighbourhood specification `ng` is practically indispensable for SGS.

9.2.2 LU Decomposition Simulation

The *LU simulation* algorithm was introduced in geostatistics by Davis (1987). It is based on the Cholesky decomposition of the variance-covariance matrix of the samples and simulation locations. The algorithm for generating realisations is as follows:

1. Compute the covariance matrix $\mathbf{C} = C(\mathbf{x}_i - \mathbf{x}_j), i, j = 1, \ldots N + M$ where N is the number of samples and M is the number of nodes in the simulation grid. This matrix can be arranged in such a way that the first DN rows and columns

represent the covariance $\mathbf{C}_s$ between sample data arranged in order of variables, where $D \geq 0$ denotes the number of variables. That is

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_s} & \mathbf{C_{sg}} \\ \mathbf{C_{sg}}^t & \mathbf{C_g} \end{bmatrix}. \tag{9.1}$$

Here the matrix $\mathbf{C_g}$ is of size DM, with entries again arranged in the order of the variables, and the matrix $\mathbf{C_{sg}}$ is the matrix of cross covariances between sample locations and grid nodes.

2. Compute the Cholesky decomposition of $\mathbf{C}$:

$$\mathbf{C} = \mathbf{A}\mathbf{A}^t, \tag{9.2}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_s} & 0 \\ \mathbf{A_{sg}}^t & \mathbf{A_g} \end{bmatrix}. \tag{9.3}$$

3. Generate a realisation by putting

$$y^\ell = \mathbf{A}_{sg}\mathbf{A}_s^{-1}Y + \mathbf{A}_g U^\ell, \tag{9.4}$$

where Y denotes the normal scores of the variables at the sample locations and U^ℓ a vector of standard normal random deviates.

In "`gmGeostats`", the Cholesky decomposition algorithm can be obtained by feeding a call to `predict.gmSpatialModel` with a method parameter specification obtained by means of the function `CholeskyDecomposition`,

```
> chl = CholeskyDecomposition(nsim, ...)
> predict(model, newdata, pars=chl)
```

9.2.3 Turning Bands

The idea behind the *turning bands simulation* method (TBS, Lantuéjoul, 2002; Chilès & Delfiner, 2012) is the reduction of a simulation in 2 or 3 dimensional space to a set of independent simulations in one-dimensional space. The independent simulations are combined into non-conditional simulations with covariance given by the covariance function of the underlying Gaussian process. To obtain conditional simulations, a kriging step is applied. To generate a non-conditional simulation using the turning bands method, proceed as follows:

1. Given a Gaussian covariance function $C(h)$ in 2 or 3 dimensional space, determine the corresponding covariance function $C_1(h)$ in one-dimensional space.

2. Determine a set of n lines $\ell_i, i = 1, \ldots, n$ with arbitrary directions in 2 or 3D space.
3. Simulate a Gaussian random function with covariance $C_1(h)$ along each line.
4. Given a point $\mathbf{x}$ on the simulation grid, determine the projection $\mathbf{x}_i$ of the point $\mathbf{x}$ on each of the lines, find the simulated values $s(\mathbf{x}_i)$ at each of the resulting points and then compute the simulated value for the grid as $s_i(\mathbf{x}) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} s_i(\mathbf{x}_i)$.
5. Repeat the process for all grid locations.

The non-conditional simulation is followed by a conditioning step where simple (co)kriging is used to interpolate the error, which is known at sample locations. The underlying assumption is that the RF $Z(\cdot)$ and the simulated RF share the same covariance function. If $Z(\mathbf{x})$ denotes the kriging true value and $S(\mathbf{x})$ the non-conditional simulation at location $\mathbf{x}$, then from

$$T(\mathbf{x}) = S(\mathbf{x}) + (Z(\mathbf{x}) - S(\mathbf{x})) \tag{9.5}$$

then at a sample location one has

$$T(\mathbf{x}_\alpha) = S(\mathbf{x}_\alpha) + (Z(\mathbf{x}_\alpha) - S(\mathbf{x}_\alpha)) = Z(\mathbf{x}_\alpha) \tag{9.6}$$

and so a cokriging of $S(\mathbf{x})$ will provide estimates of the error at non-sampled locations. These estimates are then added to the non-conditional simulated values to obtain a conditional realisation of the data (Lantuéjoul, 2002; Chilès & Delfiner, 2012).

As with the previous methods, package "`gmGeostats`" provides realisations with the turning bands algorithm if a call to `predict.gmSpatialModel` is given the `pars` argument specified by means of function `TurningBands`

```
> tb = TurningBands(nsim, ...)
> predict(model, newdata, pars=tb)
```

9.2.4 Comments

The simulation algorithms introduced in the preceding sections assume that the random function being simulated follows a multivariate normal distribution. If the regionalised composition is additive logistic normal, i.e. the log-ratio transformed data are normally distributed, then any of the algorithms mentioned may be applied without any further transformation. In this case the global mean of the data needs to be supplied, or else a shift to a mean of $\mathbf{0}$ applied which is easily reversed after the simulation. If the assumption of normality cannot be justified, then it is prudent to transform the data to multivariate normality, then apply simulation and finally

back-transform the realisations obtained. Other simulation algorithms (Chap. 10) do not require normality and may be more suitable for strong departures of joint multi-Gaussianity.

From a practical perspective, simulation via LU decomposition is only suitable if the simulation grid is small or a local simulation is sought. The main drawback of SGS is the repeated solution of (co)kriging systems which slows the simulation considerably. In this sense, TBS has advantages since the sample configuration is fixed, only one cokriging needs to be performed on the grid, as the weights are independent of what non-conditional simulation is used. This aspect makes TBS the fastest of the simulation algorithms discussed. Nevertheless, in any of these methods, the need to work in the framework of the LMC and the potential sizes of cokriging systems introduce limitations which might make the simulation of independent factors a worthwhile consideration.

9.3 Simulation of a Random Function via Univariate Simulation of PCA or MAF Factors

Techniques such as PCA or MAF can be applied to replace the variables by decorrelated factors (Desbarats & Dimitrakopoulos, 2000) which can be simulated independently, substantially reducing the computational effort. In addition, the use of PCA or MAF allows a dimension reduction by virtue of using only those factors that carry information other than pure noise. The complete workflow for this simulation approach for compositional data is as follows:

1. Apply a suitable log-ratio transformation, here ilr might be the most suitable, as no interpretation is required.
2. Transform the variables to normal scores.
3. Compute PCA or MAF factors as discussed in Chaps. 3 and 4, respectively.
4. Identify the number of factors whose spatial autocorrelation is not pure nugget.
5. For each factor, fit a suitable model describing its spatial continuity. Validate with kriging as explained in Sect. 7.2.
6. Simulate each factor based on its covariance structure.
7. Reconstitute the original normal variables from the factors.
8. Back-transform from normal scores to log-ratios.
9. Back-transform to raw compositions.

This algorithm is quite complex and requires several user decisions along the steps. For this reason it is not available as a closed form method as those explained in the preceding section. Any of the algorithms introduced earlier on could be used in step 6.

9.4 Accuracy and Precision Prior to Simulation

In Chap. 7 the validation of the variogram model and the LMC was considered. Just as in the case of (co)kriging the model of the spatial continuity to be used in the simulation should be validated prior to the actual simulation. Since the simulation methods considered in this chapter are based either on kriging or on cokriging, the validation approach from Chap. 7 can be used. We should note that the caveats and warnings from Sect. 7.5 remain pertinent. The criteria for the cross-validation output are the same as earlier on: a mean error near 0, negligible correlation between the estimation errors and estimates, no spatial autocorrelation of errors and high correlation between estimates and true values, as explained on page 136.

If the log-ratio transformed data are not normal and a normal scores transformation was applied, cross-validation should be applied to the normal scores. Even kriging estimates might be used if the scores are uncorrelated, for instance when using the MAF approach outlined in the previous section and illustrated in the next one. In these cases one must be aware that we are only testing the suitability of the normal scores model, although it is possible to back transform to the raw data space and make an assessment of the errors in that space also. More detail about validation in the back-transformed space will be given in Sect. 11.1.1.

In addition, accuracy and precision can be computed with the techniques of Sect. 7.4. As before, we are seeking a model for which the accuracy curve lies above, but ideally close to, the bisector, indicating that the conditional distribution is not overly optimistic in its prediction of the uncertainty. This is highly critical for the validated model to be acceptable for simulation. Again, for Gaussian simulation the accuracy and precision results can be derived from a cross-validation via kriging or cokriging.

Accuracy and precision may also be computed *from* the realisations, *after* the simulation process. This is discussed in Sect. 11.1.1.

9.5 Example: Windarling Data

9.5.1 Cosimulation with an LMC

In Chap. 8 (page 162) the Windarling ilr data were transformed to multivariate normality via the flow anamorphosis (FA). Our preliminary assessment showed that the normal scores are not completely spatially decorrelated, and so there are two options for the geostatistical simulation of the Windarling data. The first is to fit an LMC to the FA normal scores and perform a joint simulation. The second option is to apply a MAF transformation to the FA normal scores, followed by univariate simulation of the MAF scores and back-transformation to FA normal scores, as proposed in Sect. 9.3. We will postpone the decision by first fitting an LMC and considering its fit. The experimental direct and cross variograms are shown in Fig. 9.1 with the autofitted LMC superimposed.

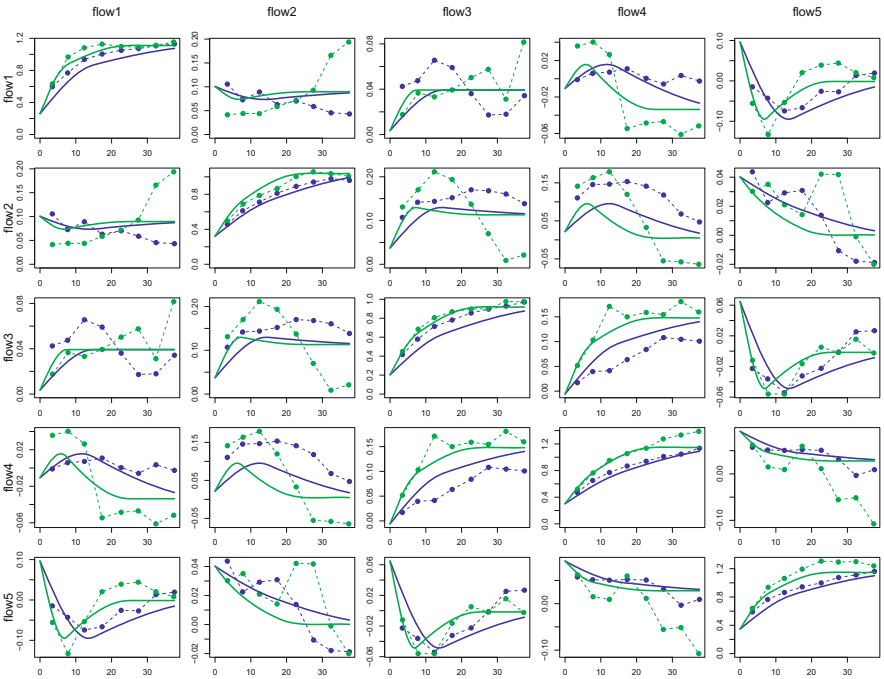

Fig. 9.1 LMC for FA normal scores of Windarling data

```
> wind.ns.gg = make.gmMultivariateGaussianSpatialModel(
+   data = data.frame(wind.ns), coords = wind.coords,
+   nmax=20, nmin=4, maxdist=20) %>% as.gstat

> wind.ns.vgAnis=variogram(wind.ns.gg,cutoff=40, width=5,
+                          alpha=c(90,180),tol.hor=45)
> wind.ns.mdAnis = vgm(nugget=0.1, psill =0.3, range=15,
+           model = "Sph", anis=c(90, 0.5))
> wind.ns.mdAnis = vgm(add.to=wind.ns.mdAnis, psill =0.6,
+                      range=50, model = "Sph", anis=c(90, 0.5))
> wind.ns.gg = fit_lmc(v=wind.ns.vgAnis, g=wind.ns.gg,
+                      model=wind.ns.mdAnis, fit.lmc=TRUE,
+                      correct.diagonal = 1.001)

> variogramModelPlot(wind.ns.vgAnis,wind.ns.gg$model)
```

While the sills of the cross-variogram models do not fit the experimental cross variograms, the shapes of the models follow those of the experimental variograms. We can therefore use this model for simulation with command `predict`.

This, nevertheless, makes no sense to do for grid points outside the mask, so we first apply the mask to the grid, and proceed to simulate

```
> wind.grid.fine.masked = setMask(wind.grid.fine,
+                                 mask = wind.mask.fine)
```

```
> wind.ns.cosim.aux = predict(wind.ns.gg,
+                         newdata=wind.grid.fine.masked, nsim=100)
```

The result depends on the class of `newdata` and there are two different formats. If `newdata` is a "`Spatial`" object ("`SpatialPoints`", "`SpatialGrid`" or "`SpatialPixels`"), then the output will be a fitting "`Spatial*DataFrame`" object with a slot `data` containing the simulated variables (100 realisations times 5 variables). If `newdata` is a "`data.frame`", then the output will be a "`data.frame`" object, this time with 502 variables, the first two being the spatial coordinates, and the rest the 100 simulations of each of the five variables; readers are invited to check that by running

```
> colnames(wind.ns.cosim.aux)
```

Package "`gmGeostats`" provides a series of functions for stacking this output in a more convenient way, as a stacked data frame. This chunk

```
> wind.ns.cosim = wind.ns.cosim.aux[,-(1:2)] %>%
+   DataFrameStack(dimnames=list(
+     loc=1:sum(wind.mask.fine), sim=1:100,
+     var=colnames(wind.ns)
+     ),
+     stackDim="sim"
+   )
```

is the way to produce the stack in the case that the output is of class "`data.frame`", while in the case of using "`Spatial`" objects, the syntax is almost the same

```
> wind.ns.cosim = wind.ns.cosim.aux@data %>%
+   DataFrameStack(
+     ...
+     )
```

The function `DataFrameStack` takes the output of a simulation, and extends it with information about the names for the rows, the columns and the stacking dimension. In the same way as "`data.frame`" has some commonalities with a matrix, a "`DataFrameStack`" has some commonalities with a 3-dimensional array. For instance, these three dimensions can be given names themselves (in the preceding code these are `loc`, `var` and `sim`). These names will be very useful later on. Stacked data frames can be also given spatial coordinates and create with them a "`SpatialPointsDataFrame`" or "`SpatialGridDataFrame`" objects

```
> wind.ilr.cosim.stacksp = SpatialPointsDataFrame(
+   coords = wind.ns.cosim.aux[,1:2], data = wind.ns.cosim
+ )
```

Another advantage of having structured the output as a stack is that one can now apply any back-transformation to the simulations in a highly transparent way, just by

```
> # compute the inverse anamorphosis
> wind.ilr.cosim.aux = gmApply(X=wind.ns.cosim,
```

```
+                              FUN=fana.wind, inv=T)
> # compute the inverse ilr
> wind.compo.cosim.aux = gmApply(X=wind.ilr.cosim.aux,
+                              FUN=ilrInv, orig=wind.acomp)
```

These lines have thus transformed our simulations first from normal score space to log-ratio space, and then from the log-ratio space to the original compositional space. Note that the function `apply` in **R** expects the second argument to be `MARGIN`, the array dimension along the function `FUN` is applied. `gmApply` is a wrapper around `apply`, particularly designed for usage with a "`DataFrameStack`" object, so that this array dimension is directly taken as the stacking dimension. With any other object `gmApply` redirects to `apply`. Hence, `gmApply` can be safely used everywhere.

If we want to display the output of any of these calculations, we need to undo the masking. We have available for that the command `unmask`,

```
> wind.compo.cosim = unmask(wind.compo.cosim.aux,
+                           mask=wind.mask.fine)
```

One can then conveniently extract any one element of the stack with the function `getStackElement` and, by binding it with the grid, plot it

```
>   out = wind.grid.fine %>%
+     cbind(getStackElement(wind.compo.cosim,1)) %>%
+     image_cokriged(ivar="Fe", legendPos = "top")
>   points(wind.coords, pch=21, col=1, cex=0.75,
+          bg=out$col[cut(wind.acomp[,"Fe"],out$breaks)])
```

Similarly, we could work with object of class "`SpatialPointsDataFrame`" or of class "`SpatialGridDataFrame`" (depending on whether `wind.grid.fine` is defined as a set of locations, or as a grid; see Sect. 6.4.1)

```
> out = wind.grid.fine %>% SpatialPointsDataFrame(
+    data=getStackElement(wind.compo.cosim,1)) %>%
+    image_cokriged(ivar="Fe", legendPos = "top")
```

Figure 9.2 shows the result of these calculations, comparing them with the original data coloured in the same scale. Readers might find it interesting to construct this map for all simulations and all variables, which can be done using the simple GUI abilities of "`manipulate`"

```
> myfun = function(i,j){
+   out = wind.grid.fine %>%
+   cbind(getStackElement(wind.compo.cosim,i)) %>%
+     image_cokriged(ivar=j, legendPos = "top")
+   points(wind.coords, pch=21, col=1, cex=0.75,
+     bg=out$col[cut(wind.acomp[,j],out$breaks)])
+ }
> manipulate(myfun(i,j),
+            i=slider(1, max = 100),
+            j=picker("Fe", "Al2O3", "SiO2", "Mn", "P", "R"))
```

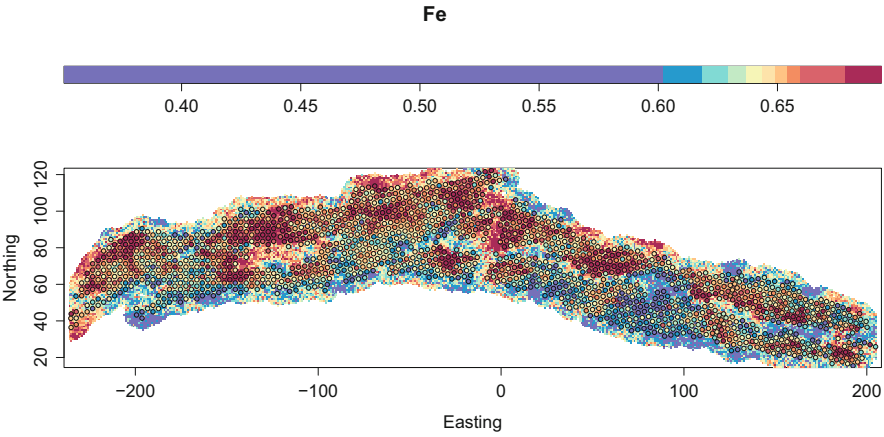

Fig. 9.2 One simulation of Fe, compared with the original values of Fe in Windarling

9.5.2 Simulation Through MAF Decomposition

An alternative to sequential Gaussian cosimulation is to use a MAF representation of the FA normal scores, so that the resulting factors are spatially decorrelated, as described in Sect. 9.3. This allows the separate simulation of each factor, to be recombined later, in the same fashion as has been used in Sect. 6.5. Here we shall use the variance–covariance matrix for the increments based on the second lag to perform the MAF transformation. The resulting factors are then stored in a data frame, and new gstat objects are constructed for each of the factors separately. These lines compute the variograms and develop the MAF factors

```
> wind.ns.vg=variogram(wind.ns.gg, cutoff=40, width=3.5)
> maf.wind.ns = Maf(data.frame(wind.ns), vg=wind.ns.vg, i=2)
```

The experimental cross and direct variograms of the MAF transformed FA normal scores can then be constructed by

```
> wind.nsmaf.gg = make.gmMultivariateGaussianSpatialModel(
+   data = maf.wind.ns$scores, coords = wind.coords )

> wind.nsmaf.vg = variogram(wind.nsmaf.gg, cutoff=40,
+                           width=3.5)
> plot(wind.nsmaf.vg, group.id=F)
```

with results shown in Fig. 9.3. The quality of the spatial decorrelation provided by the MAF factors can be appraised with function `spatialDecorrelation`

```
> par(mfrow=c(1,2))
> aux1=spatialDecorrelation(wind.nsmaf.vg,method="rdd")
> mean(aux1$gamma)

[1] 0.09259
```

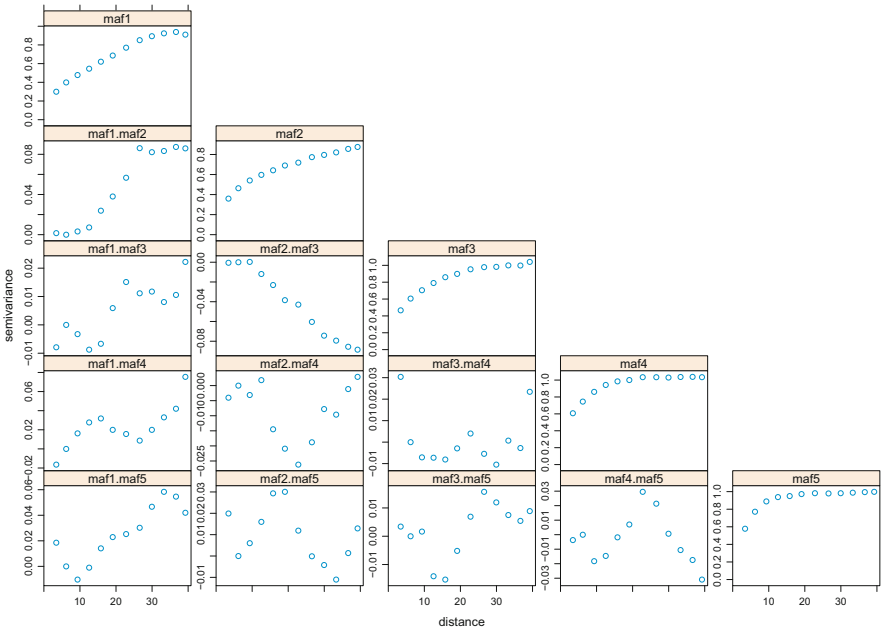

Fig. 9.3 Experimental omnidirectional direct and cross variogram of MAF transformed FA normal scores of Windarling data

```
> plot(gamma~dist,aux1, type="o",ylab="rdd")
> aux2=spatialDecorrelation(wind.nsmaf.vg,method="add")
> mean(aux2$gamma)

[1] 0.01892

> plot(gamma~dist,aux2, type="o",ylab="add")
```

Figure 9.4 shows the resulting diagrams of decorrelation, as explained in Sect. 4.5.1.

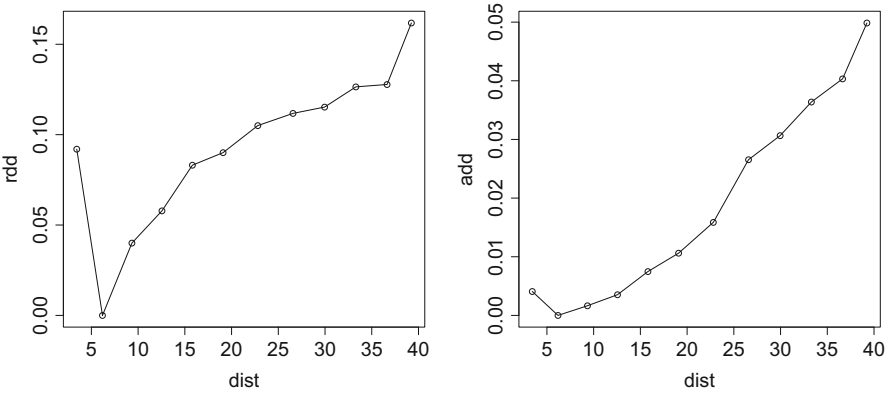

Fig. 9.4 Test of spatial decorrelation of MAF transformed FA scores: Relative (left) and absolute deviation from diagonality (right)

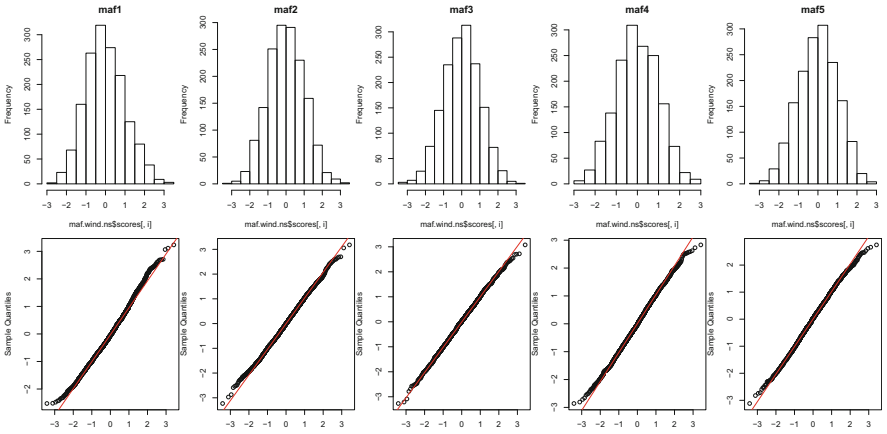

Fig. 9.5 Histograms and QQ-plots of MAF scores derived from FA normal scores of Windarling

Histograms and QQ-plots of the MAF scores in Fig. 9.5 confirm that the MAF scores are approximately normally distributed

```
> par(mfcol=c(2,5), mar=c(3,3,2,1))
> for(i in 1:5){
+   hist(maf.wind.ns$scores[,i],
+        main=colnames(maf.wind.ns$scores)[i])
+   qqnorm(maf.wind.ns$scores[,i], main="")
+   qqline(maf.wind.ns$scores[,i], col=2)
+ }
```

Spatial maps of the MAF scores are shown in Fig. 9.6, obtained with command

```
> pairsmap(data = maf.wind.ns$scores, loc = wind.coords,
+          cexrange = 1.5*c(1,1), mfrow = c(5,1),
+          foregroundcolor = NA)
```

The decorrelation of the scores does not appear to be much different from that of the FA scores, as is evident in Fig. 9.7. However, the marginal distributions of almost all (just the first factor fails) pass the normality test at confidence level 0.01:

```
> library(MVN)
> mvn(maf.wind.ns$scores,mvnTest=c("hz"),desc=FALSE)

$multivariateNormality
           Test     HZ p value MVN
1 Henze-Zirkler 0.9185  0.8028 YES

$univariateNormality
          Test  Variable Statistic   p value Normality
1 Shapiro-Wilk    maf1       0.9948  <0.001      NO
2 Shapiro-Wilk    maf2       0.9987  0.2876      YES
3 Shapiro-Wilk    maf3       0.9990  0.5473      YES
4 Shapiro-Wilk    maf4       0.9980   0.047      NO
5 Shapiro-Wilk    maf5       0.9977  0.0216      NO
```

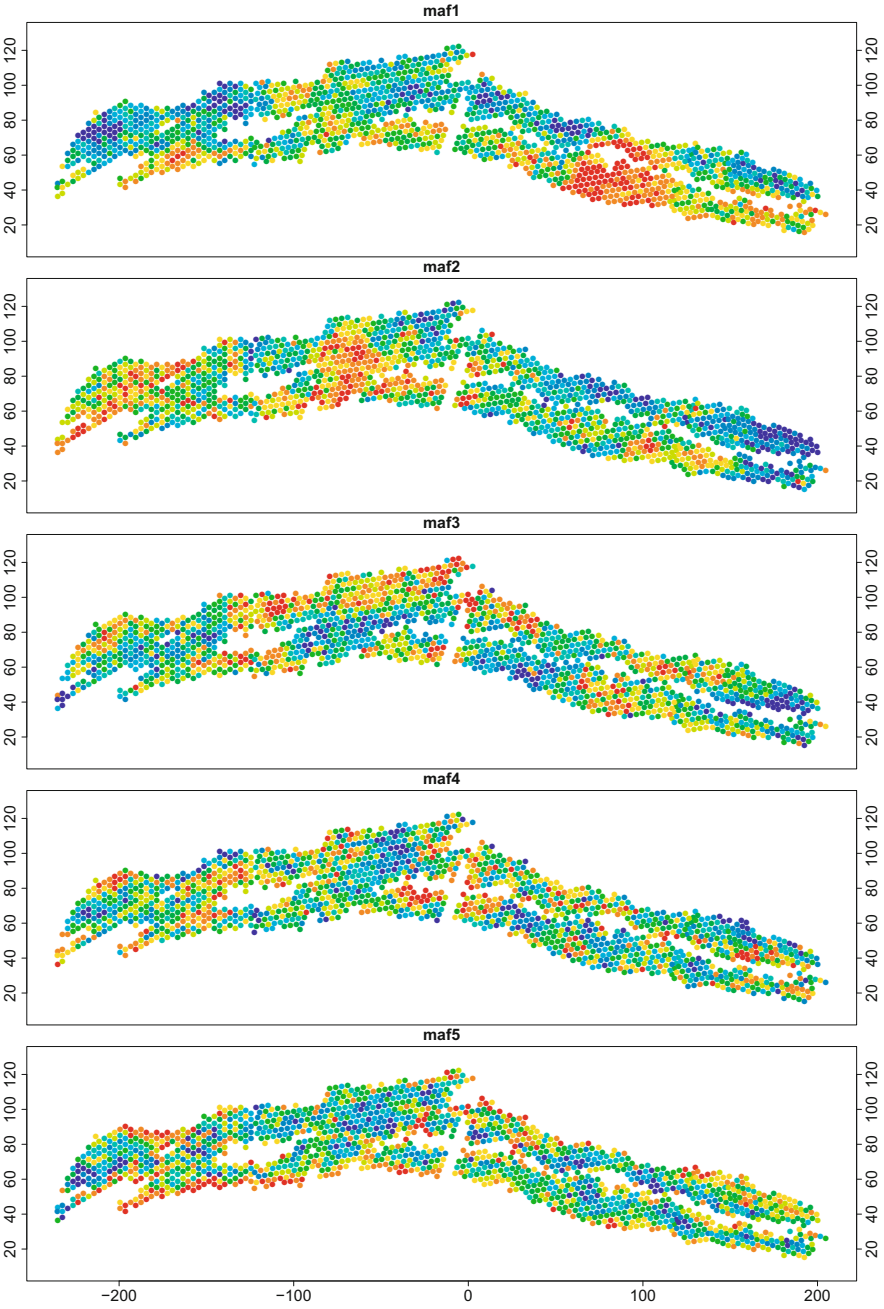

Fig. 9.6 Spatial maps of Maf Factors of FA-transformed normal scores

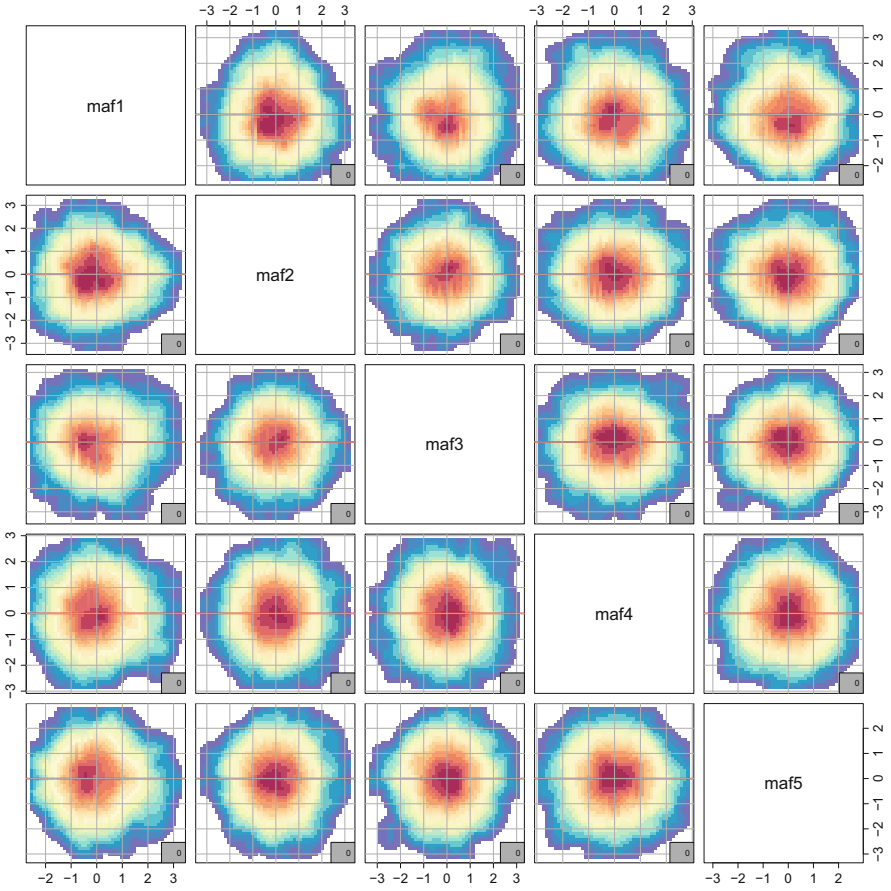

Fig. 9.7 Density plots of the MAF factors of the normal scores of Windarling

Moreover, the absolute deviation from diagonality and the relative deviation from diagonality are sufficiently close to 0 to conclude spatial decorrelation. We can therefore use univariate simulation followed by recombination of the output to generate equiprobable realisations of the Windarling data.

Individual simulations are easier to create with package "`gstat`", given that "`gmGeostats`" is designed for multivariate problems. Hence, for each individual MAF factor, we create a separate "`gstat`" object and compute its empirical variogram

```
> wind.nsmaf1.gg = gstat(id="maf1", formula=maf1~1,
+              locations = ~Easting+Northing,
+              data=data.frame(wind.coords, maf.wind.ns$scores))
> wind.nsmaf1.vg = variogram(wind.nsmaf1.gg, cutoff=30,
+                            width=3, alpha=c(0,90))
```

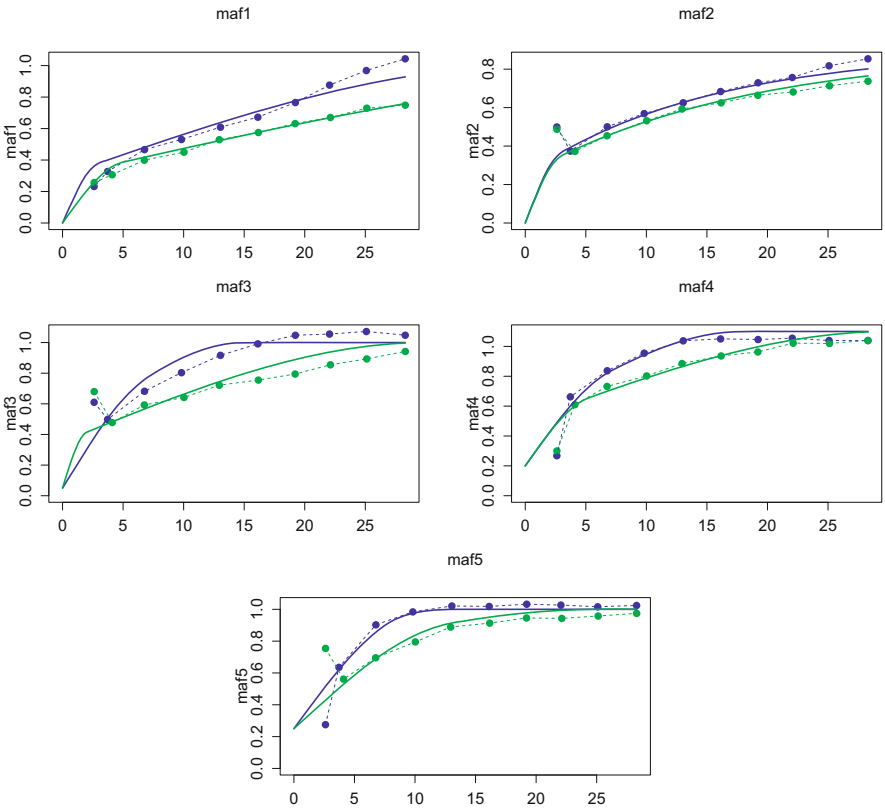

Fig. 9.8 Variogram models for the Windarling MAF transformed FA normal scores

We then model it with the tools described in Chap. 5, and add the final model to the "`gstat`" object. In our case, for the first score that would be

```
> wind.nsmaf1.md = vgm(add.to=vgm(model="Sph", nugget=0,
+                psill=0.7, range=60, anis=c(90,0.65)),
+                model="Sph", range=5, psill=0.3, anis=c(90,3/5))
> wind.nsmaf1.gg = gstat(id="maf1", formula=maf1~1,
+              locations = ~Easting+Northing,
+              data=data.frame(wind.coords, maf.wind.ns$scores),
+              model=wind.nsmaf1.md, nmax=20,nmin=4,maxdist=20)
```

The same is repeated for all five variables, in our case creating objects `wind.nsmaf2.gg` to `wind.nsmaf5.gg`. This is not shown here to save space.

The fitted variogram models compared with the directional experimental variograms of the MAF scores are shown in Fig. 9.8. These models need to be validated, with the tools we have introduced in Chap. 7, such as the function `gstat.cv`. Here

```
> wind.nsmaf5.xv= gstat.cv(wind.nsmaf5.gg)

> summary(wind.nsmaf5.xv)

   maf5.pred          maf5.var         observed
 Min.   :-2.277   Min.   :0.475   Min.   :-3.1160
```

```
 1st Qu.:-0.455   1st Qu.:0.522   1st Qu.:-0.6826
 Median :-0.010   Median :0.524   Median : 0.0211
 Mean   :-0.011   Mean   :0.542   Mean   : 0.0000
 3rd Qu.: 0.433   3rd Qu.:0.549   3rd Qu.: 0.6911
 Max.   : 2.279   Max.   :0.837   Max.   : 2.7478
    residual          zscore            fold
 Min.   :-3.266   Min.   :-4.522   Min.   :   1
 1st Qu.:-0.455   1st Qu.:-0.624   1st Qu.: 396
 Median : 0.010   Median : 0.014   Median : 792
 Mean   : 0.011   Mean   : 0.009   Mean   : 792
 3rd Qu.: 0.458   3rd Qu.: 0.623   3rd Qu.:1187
 Max.   : 2.399   Max.   : 3.186   Max.   :1582
    Easting          Northing
 Min.   :-235.3   Min.   : 15.5
 1st Qu.:-112.4   1st Qu.: 53.0
 Median : -19.7   Median : 68.0
 Mean   : -15.4   Mean   : 68.7
 3rd Qu.:  84.2   3rd Qu.: 85.8
 Max.   : 204.8   Max.   :122.2
```

a summary for the cross-validation results for the fifth MAF model is shown as an example (to save space). The results of all five cross-validations are shown in Fig. 9.9.

From these results, it can be concluded that the variogram models of the MAF variables are suitable for simulation, and we will use them to generate 100 realisations via sequential Gaussian simulation. For instance, for the first MAF factor that would be

```
> wind.nsmaf1.sim = predict(wind.ns.maf.gg1,
+         newdata=wind.grid.fine.masked, nsim=100)

> dim(wind.nsmaf1.sim)

[1] 24266   102
```

Realisations for factors maf2 to maf5 can be generated similarly, creating data frames with the coordinates of each location and 100 simulations of the corresponding factor, in objects `wind.nsmaf2.sim` to `wind.nsmaf5.sim`. We can then construct a data frame stack of the simulations only (that is, removing the first two columns that contain the coordinates, or else by taking the `@data` slot if these were "`Spatial`" objects). In the LMC-based approach we have seen that "`DataFrameStack`" objects can be seen as analogous to arrays, and they were constructed from an array. Here it is simpler to understand them as lists of data frames, and easy to construct as such. For this, it is practical to first extract the dimnames of the simulations for the first maf, as these will serve as common names

```
> dn =dimnames(wind.nsmaf1.sim[,-(1:2)])
> names(dn) = c("loc","sim")
> dn$var = colnames(maf.wind.ns$scores)
> wind.nsmaf.sim.aux = list(
+   wind.nsmaf1.sim[,-(1:2)],
+   wind.nsmaf2.sim[,-(1:2)],
+   wind.nsmaf3.sim[,-(1:2)],
+   wind.nsmaf4.sim[,-(1:2)],
+   wind.nsmaf5.sim[,-(1:2)]
```

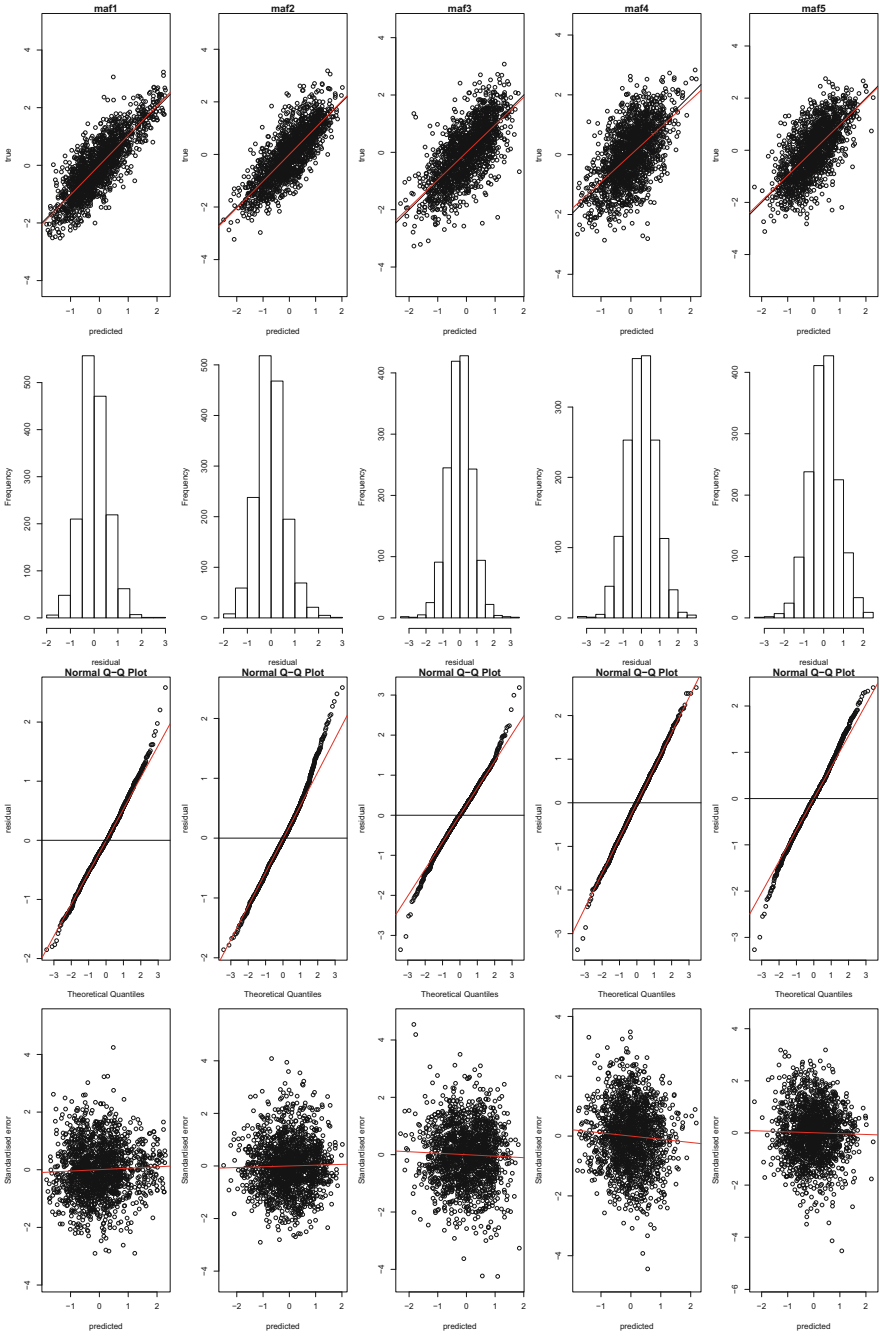

Fig. 9.9 Cross-validation results for models of MAF transforms of normal scores for Windarling data

```
+ ) %>% DataFrameStack(
+   stackDimName = "var", Dimnames=dn
+ )
```

The results now need to be back-transformed three times: first from MAF variables to normal scores, then from normal scores to log-ratios and finally to compositions. Each of these transformations must be done in parallel for each simulation. For this, function `gmApply.DataFrameStack` will be used:

```
> # declare dimension "sim" to be the stacking dimension
> stackDim(wind.nsmaf.sim) = "sim"
> wind.ns.sim.aux = wind.nsmaf.sim.aux %>%
+   gmApply(FUN=predict, object=maf.wind.ns)
> wind.ilr.sim.aux = wind.ns.sim.aux %>%
+   gmApply(FUN=fana.wind, inv=T)
> wind.compo.sim.aux =  wind.ilr.sim.aux %>%
+   gmApply(FUN=ilrInv, orig=wind.acomp)
```

Each of these transformations requires its extra arguments, as shown above. Finally, the realisations need to be unmasked

```
> wind.compo.sim = unmask(wind.compo.sim.aux,
+                         mask =wind.mask.fine)
```

and results can be plotted as in the preceding section

```
>   out = wind.grid.fine %>%
+     cbind(getStackElement(wind.compo.sim,1)) %>%
+     image_cokriged(ivar="Fe", legendPos = "top")
>   points(wind.coords, pch=21, col=1, cex=0.75,
+          bg=out$col[cut(wind.acomp[,"Fe"],out$breaks)])
```

Figure 9.10 shows the result of these calculations; compare them with Fig. 9.2 to see how well the MAF decomposition worked. Readers are recommended to adapt the manipulate code chunk on page 175 and evaluate the variability of the simulations.

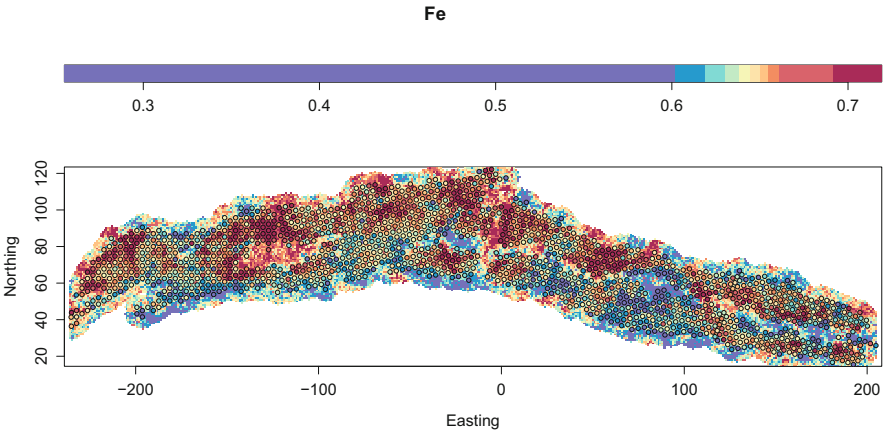

Fig. 9.10 One simulation of Fe via MAF decomposition, compared with the original values of Fe in Windarling

Problems

9.1 Simulation via LMC of the Tellus Subcomposition in Subset

(a) Use the flow-anamorphosis parameters derived in Problem 8.3 to transform the Tellus subcomposition {MgO, Fe_2O_3, CaO, Al_2O_3, Rest} within the subset (identified by Flag=1) to multivariate normality.
(b) Fit an LMC to the normal scores, validate it and use them to generate 25 realisations of the Tellus subcomposition.

9.2 Simulation via MAF of the Tellus Subcomposition in Subset

(a) Use the flow-anamorphosis parameters derived in Problem 8.3 to transform the Tellus subcomposition {MgO, Fe_2O_3, CaO, Al_2O_3, Rest} within the subset (identified by Flag=1) to multivariate normality.
(b) Calculate MAF factors of the normal scores, fit variogram models and validate them.
(c) Generate 25 realisations of the Tellus subcomposition based on the MAF factors.

9.3 Simulation via MAF of the NGSA Major Elements

(a) Apply the flow anamorphosis with parameters as determined in Problem 8.4 to determine normal scores for the major elements selected with `REGION=="EAST"` and `CODE=="Bc"`.
(b) Calculate MAF factors of the normal scores, fit variogram models and validate them.
(c) Generate 25 realisations of this NGSA subcomposition based on the MAF factors on a grid with cells of size 5 km by 5km.

References

Chilès, J. P., & Delfiner, P. (2012). *Geostatistics - Modeling spatial uncertainty* (2 ed., 699 pp.). Hoboken, New Jersey (USA): Wiley.

Davis, M. W. (1987). Production of conditional simulations via the LU decomposition of the covariance matrix. *Mathematical Geology, 19*(2), 91–98.

Desbarats, J., & Dimitrakopoulos, R. (2000). Geostatistical simulation of regionalised pore-size distributions using min/max autocorrelation factors. *Mathematical Geology, 32*(8), 919–942.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation* (483 pp.). Applied Geostatistics Series. New York, NY, USA: Oxford University Press.

Journel, A. G., & Huijbregts, C. J. (1978). *Mining geostatistics* (600 pp.). London, UK: Academic Press.

Lantuéjoul, C. (2002). *Geostatistical simulation: Models and algorithms* (256 pp.). New York, NY (USA): Springer.

Chapter 10
Compositional Direct Sampling Simulation

Hassan Talebi, Ute Mueller, and Raimon Tolosana-Delgado

Abstract In many instances it is desirable to capture more than the first two moments of the data when exploring their variability. In this chapter direct sampling simulation for compositional data is introduced, which explicitly incorporates multiple-point statistics in the simulation.

10.1 Introduction

Earth science data typically show complex statistical and spatial patterns and non-linear relationships, like the complex spatial patterns of river flood plains and deltas; the three-dimensional conical and faulted structure of porphyry copper deposits or the inter-related dynamics of topography, geomorphology and climate. Traditional two-point, trans-Gaussian[1] geostatistical algorithms based on variograms are rather limited models for describing such complex physical realities. Multiple-point simulation (MPS) methods use training images or data and are designed to improve the physical realism of the predictions and uncertainty models under these complex geometric controls (Mariethoz & Caers, 2015). In recent years, many different MPS algorithms have been developed, which can run cosimulation of multivariate data. However, these algorithms need to be (often, just slightly) adapted for the simulation of compositional data. There are only few studies where the compositional nature of input data was considered within the MPS framework (e.g. Boogaart et al. (2018); Talebi et al. (2019)). In this chapter an adaptation of the direct sampling algorithm (Mariethoz & Renard, 2010) is considered to account for a consistent and affine equivariant simulation of compositional information. Odds are in favour that the same adaptation strategy will work for many other MPS algorithms.

CSIRO Deep Earth Imaging FSP, Australia, Hassan.Talebi@csiro.au

[1] Gaussian methods are those explained in all previous chapters; trans-Gaussianity refers to the property of a random function to become tractable by means of a point-wise transformation, as the one described in Chap. 8.

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_10

10.2 Compositional Direct Sampling Simulation

In what follows we assume that the compositional data have been migrated to nodes of the grid on which the compositional random function is to be simulated and that a training image (TI) with the same resolution is available to extract patterns that will be used during the simulation. To generate a realisation, a random path is defined visiting all the unsampled locations on the grid. Given a grid node $\mathbf{x}$ whose composition is to be simulated, a *data event* for $\mathbf{x}$ is defined to be the set $\mathscr{E}(\mathbf{x}) = \{\mathbf{z}(\mathbf{x}_1), \mathbf{z}(\mathbf{x}_2), \ldots \mathbf{z}(\mathbf{x}_N)\}$ consisting of the compositional vectors at the N nearest locations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ to $\mathbf{x}$. For each $i = 1, \ldots, N$ we denote by $\mathbf{h}_i = \mathbf{x}_i - \mathbf{x}$ the difference vector between the ith location in the data event $\mathscr{E}(\mathbf{x})$ and $\mathbf{x}$. Then a location $\mathbf{x}^{TI}$ is drawn randomly from the training image. The data event for $\mathbf{x}^{TI}$ is then defined to be the set $\mathscr{E}^{TI}(\mathbf{x}^{TI}) = \left\{\mathbf{z}(\mathbf{x}_1^{TI}), \mathbf{z}(\mathbf{x}_2^{TI}), \ldots \mathbf{z}(\mathbf{x}_N^{TI})\right\}$ where $\mathbf{x}_i^{TI} = \mathbf{x}^{TI} + \mathbf{h}_i$, $i = 1, \ldots, N$. This ensures that the data event in the training image has the same geometry as $\mathscr{E}(\mathbf{x})$.

The distance between the data events $\mathscr{E}(\mathbf{x})$ and $\mathscr{E}^{TI}(\mathbf{x}^{TI})$ is defined as

$$d_{DSA}(\mathscr{E}(\mathbf{x}), \mathscr{E}^{TI}(\mathbf{x}^{TI})) = \frac{1}{d^{max}} \sum_{n=1}^{N} w_n d_A(\mathbf{z}(\mathbf{x}_n), \mathbf{z}(\mathbf{x}_n^{TI})), \tag{10.1}$$

where d_A denotes the compositional distance defined in Eq. (2.7) in Chap. 2, the normalising factor d^{max} is given by

$$d^{max} = \max\left\{d_A(\mathbf{z}(\mathbf{x}^{TI}), \mathbf{z}(\mathbf{y}^{TI})) : \mathbf{x}^{TI}, \mathbf{y}^{TI} \in TI\right\}$$

and $w_n, n = 1, \ldots, N$ denote positive weights with $\sum_{n=1}^{N} w_n = 1$.

The distance $d_{DSA}(\mathscr{E}(\mathbf{x}), \mathscr{E}^{TI}(\mathbf{x}^{TI}))$ and the training data event $\mathscr{E}^{TI}(\mathbf{x}^{TI})$ are stored. A new location $\mathbf{x}_*^{TI}$ in the training set is drawn at random and the process is repeated for the new location. If $d_{DSA}(\mathscr{E}(\mathbf{x}), \mathscr{E}^{TI}(\mathbf{x}_*^{TI})) < d_{DSA}(\mathscr{E}(\mathbf{x}), \mathscr{E}^{TI}(\mathbf{x}^{TI}))$, the record is updated by replacing the data event $\mathscr{E}^{TI}(\mathbf{x}^{TI})$ by $\mathscr{E}^{TI}(\mathbf{x}_*^{TI})$ and updating the distance. This process continues until either the distance $d_{DSA}(\mathscr{E}(\mathbf{x}), \mathscr{E}^{TI}(\mathbf{x}^{TI}))$ is less than a pre-defined threshold t or the fraction of the training image scanned exceeds a pre-defined fraction f without identification of a training data event with distance from $\mathscr{E}(\mathbf{x})$ less than t. In either case the composition $\mathbf{z}(\mathbf{x}^{TI})$ associated with the closest pattern obtained so far is assigned to $\mathbf{z}(\mathbf{x})$.

As the simulation progresses, the spatial range of the data event decreases, as the spatial density of the informed nodes increases. The natural variation of data events during the simulation makes the algorithm flexible in terms of capturing structures and patterns of different scales from the training image. It should be noted that at each location the entire compositional vector is simulated at the same time.

As noted in Chap. 2, the compositional distance can be re-written as the Euclidean distance between the clr- or ilr-transformed compositions (Eq. 2.11), so

that direct sampling simulation can be applied directly to the clr- or ilr-transformed compositions of the sample set and the training image. Since the ilr-transformation reduces the dimension of the data by 1, it is preferable to the clr-transform. Any ilr is valid, and results do not depend on which one is used because of affine equivariance.

10.3 Comments

10.3.1 Implementation

The version of the direct sampling algorithm considered here is designed to generate realisations of a compositional random function based on the use of full compositional information at the sample locations and within the training image. The transformation to log-ratios is performed as a first step in the algorithm, so that the full compositional information in the sample needs to be provided. Further, the sample data need to be migrated to the simulation grid prior to simulation. Such migration can be achieved, for instance, with an appropriate usage of command `SpatialPixelsDataFrame`: readers are invited to check the help page of this function, and the interplay between its arguments `points`, `tolerance` and `grid`.

Package "`gmGeostats`" provides a direct sampling algorithm for compositional data in the function `gsi.DS4CoDa`. This command uses a functionality from package "`FNN`", required for its fast nearest neighbour search in the direct sampling. Such internal functions should not be called directly; instead it is recommended to adopt a stepwise approach.

First create the training image (a "`SpatialGridDataFrame`" or "`SpatialPixelsDataFrame`", depending on masking)

```
> ti.comp = SpatialGridDataFrame(grid, data)
```

Define the geostatistical data-object model

```
> mps.comp = make.gmCompositionalMPSSpatialModel(data, V="ilr",
+                                                model=ti.comp)
```

Define the grid for simulation (again "`SpatialGrid`" or "`SpatialPixels`", depending on masking),

```
> nwd = SpatialGrid(grid)
```

Then define the direct simulation parameter object

```
> ds.pars = DSpars(nsim, scanFraction, patternSize, gof)
```

Finally run the algorithm by means of a `predict` call

```
> DSsims = predict(mps.comp, newdata=nwd, pars=ds.pars)
```

The reason is that through this structured procedure, consistency checks can be run for each step. In the next sections the meaning and exact usage of these parameters and functions will be discussed and illustrated.

10.3.2 Parameter Choices

The most critical input parameters are the maximum fraction f of the training image to be scanned (parameter `scanFraction`), the number N of nodes in the spatial patterns (parameter `patternSize`) and the distance threshold t (parameter `gof`, a value between 0 and 1). Choosing a value for f less than one increases the speed of algorithm and decreases the chance of verbatim copy. Choosing a large value for t increases the speed of algorithm, improves the reproduction of the statistical patterns, but deteriorates spatial pattern reproduction. Similarly, the more nodes in the pattern, the better the quality of the simulation. Good references for parameter selection in DS and other MPS techniques are Meerschman et al. (2013), Talebi et al. (2019) and Bananijar et al. (2019). In particular, Bananijar et al. (2019) propose an optimisation program to select the parameters optimally for a range of MPS methods.

10.3.3 Training Image Generation

Large compositional training images are necessary to accurately reproduce the spatial patterns and to model the spatial uncertainties via the compositional direct sampling algorithm (Emery & Lantuéjoul, 2014). Methods for constructing compositional training images include the use of moving windows to compute the proportions of multiple classes of interest or upscaling discrete raster data (e.g. geology, soil type and land use map). Densely sampled geochemical data can also be assigned to a regular grid to form a compositional training image. High-resolution airborne radiometric data (potassium (%), equivalent thorium (ppm) and equivalent uranium (ppm)) is another possibility to obtain such training images for certain compositional variables.

10.4 Example: Direct Sampling Simulation of a Subcomposition of the Tellus Data

The geochemical components considered in this example are Al_2O_3, Fe_2O_3, MgO and CaO and have already been used to illustrate estimation via MAF in Sect. 6.5. These four variables were chosen to capture the contrasting geology of Northern Ireland: detritic clastic sedimentary rock and granites are captured by high values of Al_2O_3; Fe_2O_3 is enriched in basalts and certain continental sedimentary rock; high MgO is a strong characteristic of basalts and dolomitic rocks, while CaO represents limestones and might assist in distinguishing between magmatic rocks. The sum of the masses of all other chemical components not considered in this analysis will be stored as a rest variable, called Rest.

10.4.1 Training Image and Regridding

As a first step the sample data are extracted from the full Tellus regional data set and the training image is prepared.

```
> setwd("YOUR_WORKING_DIRECTORY")
> getTellus(cleanup = TRUE, TI = TRUE)
```

The spatial coordinates and the selected subcompositions are stored in a "SpatialPointsDataFrame" object. The variable *Flag* indicates the subsample of the Tellus data to be used as conditioning data. The first 6 rows of the sample coordinates and the sample subcomposition are displayed with the command `head`.

```
> tellus=TellusASoil
> tellusS.compo = dplyr::filter(tellus, Flag==1) %>%
+   dplyr::select(MgO,Al2O3,CaO,Fe2O3)
> tellusS.compo$Rest = 100 - rowSums(tellusS.compo)
> tellusS.coords = dplyr::filter(tellus,Flag==1) %>%
+   dplyr::select(EASTING:NORTHING)
> head(tellusS.coords)

  EASTING NORTHING
1  222953   350113
2  237448   345172
3  226465   356673
4  226272   346797
5  227575   343849
6  222979   352924

> head(tellusS.compo)

  MgO Al2O3  CaO Fe2O3 Rest
1 1.6  11.1 0.70  3.76 82.8
2 0.9   9.3 0.78  2.05 87.0
3 2.4  13.5 0.56  6.04 77.5
4 1.3  10.1 0.84  2.26 85.5
5 1.3   9.3 0.69  3.20 85.5
6 1.5  11.5 0.64  3.40 83.0

> tellusS.spdf = SpatialPointsDataFrame(coords = tellusS.coords,
+                                       data=tellusS.compo)
```

Next a working copy of the training image needs to be loaded as well

```
> tellus.TI= Tellus_TI

> head(tellus.TI)

      EASTING NORTHING MgO Al2O3 CaO Fe2O3 Rest Mask
13159  188183   452272  NA    NA  NA    NA   NA    0
13030  188183   450872  NA    NA  NA    NA   NA    0
12901  188183   449472  NA    NA  NA    NA   NA    0
12772  188183   448072  NA    NA  NA    NA   NA    0
12643  188183   446672  NA    NA  NA    NA   NA    0
12514  188183   445272  NA    NA  NA    NA   NA    0
```

This training image was generated by migrating the Tellus sample data of interest into a regular grid. The variables in the training image are the grid coordinates, the subcomposition and an indicator variable called Mask, which indicates grid locations within the study region. The grid parameters are 129 nodes in the E-W direction, 103 nodes in the N-S direction and a cell-size of 1400 m. The grid origin is (188183, 309472).

```
> easting = unique(tellus.TI$EASTING)
> northing = unique(tellus.TI$NORTHING)
> length(easting)

[1] 129

> length(northing)

[1] 103

> range(easting)

[1] 188183 367383

> range(northing)

[1] 309472 452272

> mean(diff(easting))

[1] 1400

> mean(diff(northing))

[1] -1400
```

With this information we can define the grid topology for the study.

```
> x0 = c(min(easting), min(northing))
> Dx = abs(c(mean(diff(easting)), mean(diff(northing))))
> nn = c(length(easting), length(northing))
> tellus.gt = GridTopology(cellcentre.offset = x0,
+                          cellsize = Dx, cells.dim = nn)
```

The value of Mask is 1, if a location is outside of the simulation region and 0 if it is within. Hence, it almost directly specifies a mask to use with `setMask` as was introduced in Chap. 6. The only problem might be related to the ordering of the grid : while the given training image starts at the upper left corner and runs faster eastwards then southwards, grid topologies from package "`sp`" have their origin in the same upper left corner but run faster southwards than eastwards. Indeed, the whole point is a matter of convention, and different conventions exist. To avoid messing up these things, and given that we do have the real coordinates of each point of the training image grid, it is convenient to allow "`sp`" to reorder the points on itself, like this

```
> colnames(tellus.TI)

[1] "EASTING"  "NORTHING" "MgO"      "Al2O3"    "CaO"
[6] "Fe2O3"    "Rest"     "Mask"
```

```
> tellus.TI.aux =
+  SpatialPixelsDataFrame(points = SpatialPoints(tellus.TI[,1:2]),
+                 grid=tellus.gt, tolerance = sqrt(sum(Dx^2))/2,
+                 data=tellus.TI)  %>%
+   as("SpatialGridDataFrame")
```

The tolerance is taken as half the diagonal size of the pixels. Then we just need to extract the mask (recasting it to logical and inverting it) and the variables of interest, both as columns of the slot `@data`

```
> tellus.mask = !as.logical(tellus.TI.aux@data$Mask)
> tellus.TI.spdf = tellus.TI.aux@data %>%
+   dplyr::select(MgO:Rest) %>%
+   SpatialGridDataFrame(grid=tellus.gt)
```

Alternatively, package "`gmGeostats`" provides a way to encode the grid ordering and change between grids. Orderings are encoded by lists of two elements: `refpoint`, controlling which is the first point of the grid; and `cycle`, controlling which dimension runs faster; i.e. in this case first Northing, then Easting:

```
> oo = list(refpoint="topleft", cycle=2:1)
```

Additionally the command `sortDataInGrid` allows reordering gridded data to another possible grid ordering (see the help of this command for more information)

```
> tellus.TI = sortDataInGrid(tellus.TI, grid = tellus.gt,
+                 orderIn=oo, orderOut = gridOrder_gstat(G=2) )
> head(tellus.TI)
```

	EASTING	NORTHING	MgO	Al2O3	CaO	Fe2O3	Rest	Mask
13159	188183	452272	NA	NA	NA	NA	NA	0
13160	189583	452272	NA	NA	NA	NA	NA	0
13161	190983	452272	NA	NA	NA	NA	NA	0
13162	192383	452272	NA	NA	NA	NA	NA	0
13163	193783	452272	NA	NA	NA	NA	NA	0
13164	195183	452272	NA	NA	NA	NA	NA	0

The code `gridOrder_sp(G=2)` is just a shortcut for the way of ordering a grid in "`sp`" (and "`gstat`"). There is as well a `gridOrder_array` for an ordering from the bottom left corner, first northwards then eastwards (then upwards in 3D).

Spatial maps of the training images for the subcomposition Al_2O_3, Fe_2O_3, MgO, CaO and Rest are shown in Fig. 10.1, and obtained with this command

```
> myfun = function(j){
+   tellus.TI.spdf %>% image_cokriged(ivar=j, legendPos = "top",
+                                     legendPropSpace = 0.12) }

> sapply(c("MgO", "Al2O3", "CaO", "Fe2O3", "Rest"), myfun)
```

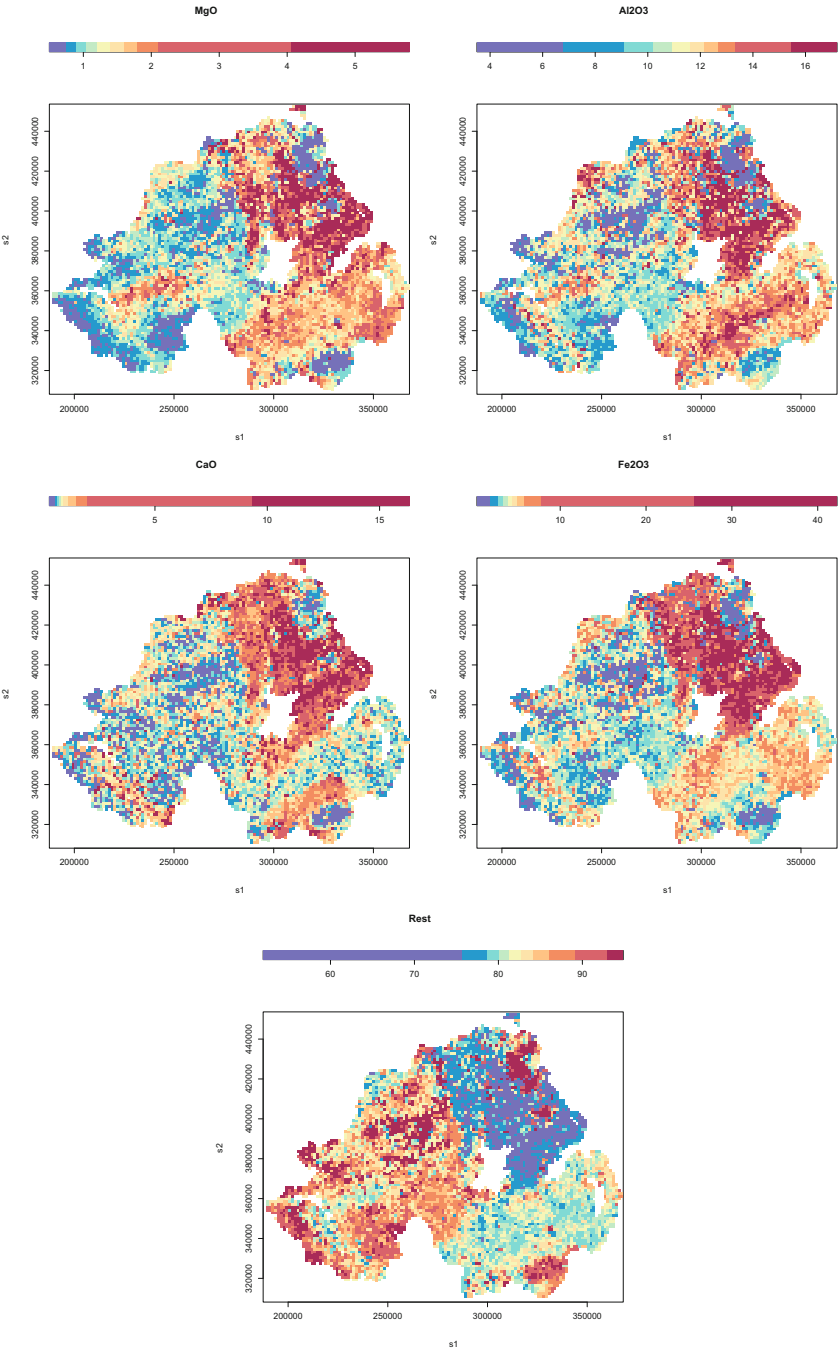

Fig. 10.1 Compositional training image generated from Tellus soil geochemical data

10.4.2 Conditional Spatial Model

The simulation grid will be constructed using the same dimensions and mask as the training image

```
> tellusS.cgrid = SpatialGrid(grid=tellus.gt) %>%
+   setMask(tellus.mask)
> class(tellusS.cgrid)
```

```
[1] "SpatialPixels"
attr(,"package")
[1] "sp"
```

The conditioning data need to be migrated to the simulation grid. This is done directly by the function `SpatialPixelsDataFrame` providing the grid topology and an appropriate tolerance

```
> tellusS.data = SpatialPixelsDataFrame(
+   points=tellusS.coords, data = tellusS.compo,
+   tolerance = sqrt(sum(Dx^2))/2, grid = tellus.gt)
```

The tolerance is half the diagonal size of the pixels, as before. To double-check that the conditioning data have been properly migrated it is customary to plot maps of the conditioning data (Fig. 10.2),

```
> myfun = function(j){
+   bks = quantile(tellus.TI[,j], prob=(0:10)/10, na.rm=T)
+   tellusS.data %>% image_cokriged(ivar=j, legendPos = "top",
+                          breaks = bks, legendPropSpace = 0.12)
+ }
```

```
> sapply(c("MgO", "Al2O3", "CaO", "Fe2O3", "Rest"), myfun)
```

Now we can create the conditional spatial model

```
> tellusS.cmodel = make.gmCompositionalMPSSpatialModel(
+   data=tellusS.data, V = "ilr", model = tellus.TI.spdf )
```

10.4.3 Simulation

Once the spatial model and the simulation grid are available, the simulation parameters need to be defined. Here the maximum number n of nodes in a data event is set to 16, the distance threshold t is set to 0.05 and the maximum fraction f of the training image to be scanned is set to 0.75

```
> tellus.ds.pars =
+   DSpars(nsim=25, scanFraction=0.75, patternSize=16, gof=0.05)
```

The simulation is then performed by calling the function `predict`

```
> tellusS.dsim = predict(tellusS.cmodel, newdata=tellusS.cgrid,
+                        pars=tellus.ds.pars)
```

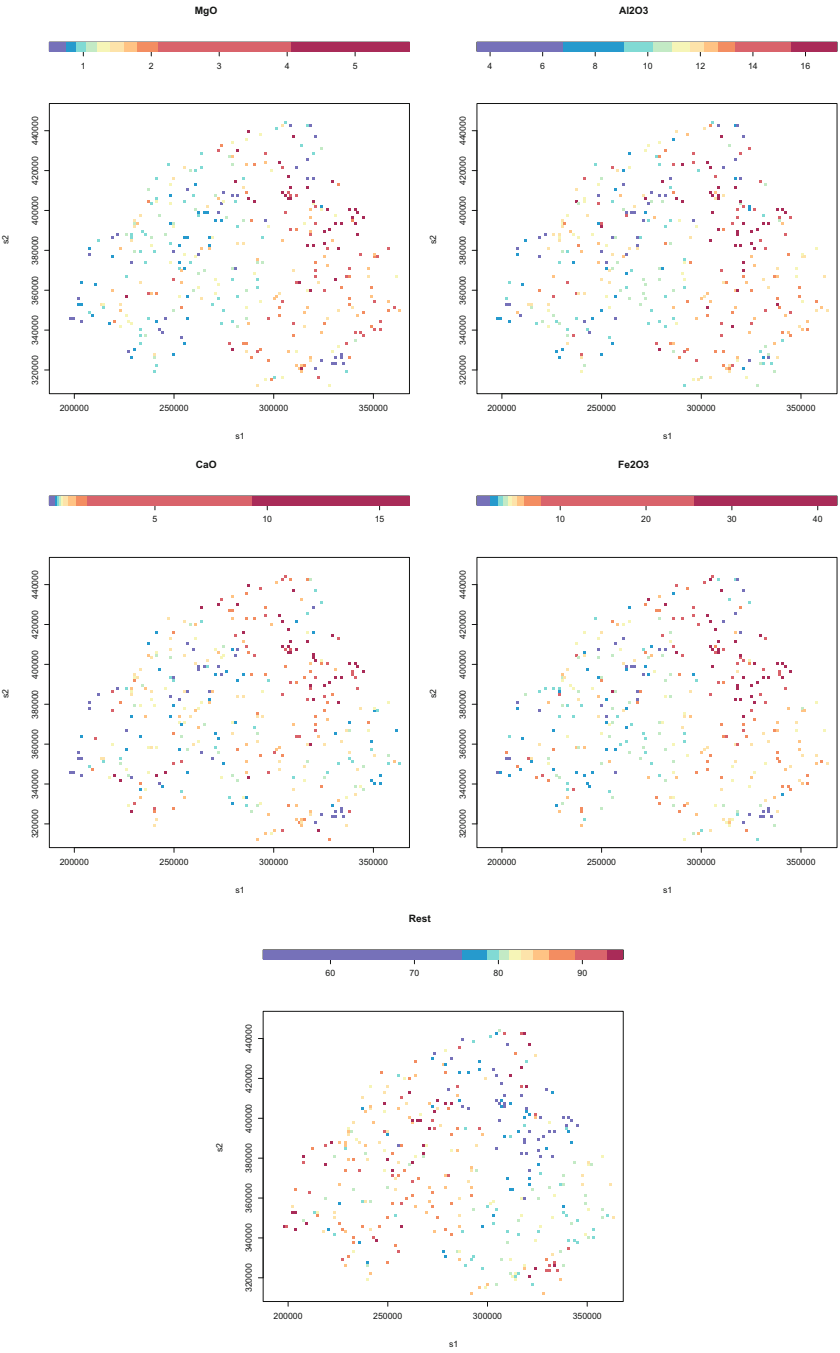

Fig. 10.2 Conditioning compositional data from Tellus XRF data set

It should be noted that the simulation is slow, so it is advisable to store the simulation results in an "`*.RData`" file prior to the evaluation.

```
> save(tellusS.cmodel, tellusS.cgrid, tellus.ds.pars,
+                   file="DS4CoDaTellusInput.RData")
> save(tellusS.dsim, file="DS4CoDaTellusSim.RData")
```

The simulated results are automatically stored in a "`SpatialGriDataFrame`", whose slot `data` is an "`DataFrameStack`" object, as already discussed in Chap. 9. Remember that this is analogous to both a data frame and to a three-dimensional array. As a data frame, the object has as many rows as nodes in the simulation grid, and the number of columns is equal to the product of the number of variables and the number of simulations.

```
> class(tellusS.dsim)

[1] "SpatialGridDataFrame"
attr(,"package")
[1] "sp"

> class(tellusS.dsim@data)

[1] "DataFrameStack" "data.frame"

> dim(tellusS.dsim)

[1] 13287   125

> getGridTopology(tellusS.dsim)

                       X1     X2
cellcentre.offset 188183 309472
cellsize            1400   1400
cells.dim            129    103
```

As a three-dimensional array, the first dimension runs through the nodes of the simulation grid, the second dimension through the simulated components and the third dimension through the realisations,

```
> sapply(dimnames(tellusS.dsim), length)

  loc   var   sim
13287     5    25
```

i.e. the stacking dimension is the third one

```
> stackDim(tellusS.dsim@data)

[1] "sim"
```

To check that the realisations match the features of the input data, one realisation is picked at random with function `getStackElement` and spatial maps of the attributes are generated (Fig. 10.3),

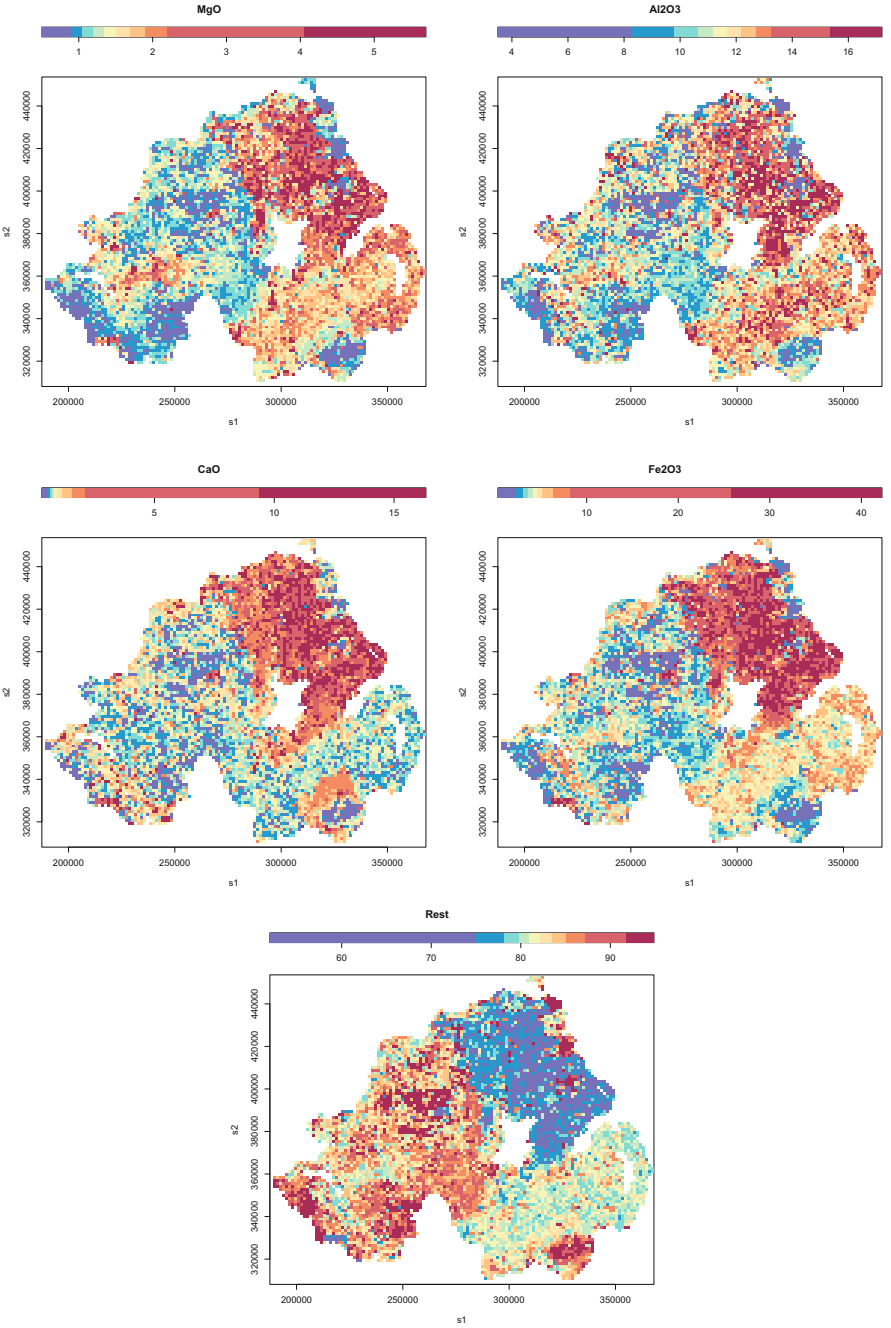

Fig. 10.3 A randomly selected realisation of the conditional simulation via compositional direct sampling algorithm

```
> tellus.1 = cbind(coordinates(tellusS.dsim),
+                       100*getStackElement(tellusS.dsim@data,1))
> myfun = function(j){
+   out = tellus.1 %>% image_cokriged(ivar=j, legendPos = "top",
+                     legendPropSpace = 0.12)
+ }

> sapply(c("MgO", "Al2O3", "CaO", "Fe2O3", "Rest"), myfun)
```

A comparison of the maps in Figs. 10.3 and 10.1 reveals the power of the technique in terms of reproducing complex spatial patterns. In the example considered here, the training image consisted of a gridded version of the densely sampled Tellus geochemical data, and although the sample size of the sample was small compared to the full data set (5% of the data were used for conditioning), the main features of the subcomposition were captured adequately. Adherence to compositional principles was guaranteed by pasting the entire composition identified as best. In some sense this example is overoptimistic, because the training image and the conditioning data were extracted from the same set. We will see that this has consequences in the evaluation later on in Sect. 11.3, page 226.

10.5 Example: Direct Sampling Simulation of Windarling East Data Based on Windarling West

A more realistic (and possibly useful) application arises when compositional information needs to be simulated at unsampled locations in a grid from known compositional information in a different but denser grid, for example available from a mined out deposit of similar style. The idea will be illustrated with the Windarling data set, taking all samples from Windarling West, but only the sample from Windarling East:

```
> windDS.subset = (windarling$West+windarling$Sample.East)==1
> windDS.gcoords = wind.coords[windDS.subset,]
> windDS.gcompo = wind.compo[windDS.subset,]
> plot(windDS.gcoords, asp=1)
```

The information in the western part of the bench is at a much denser resolution than in the east as shown in Fig. 10.4. The workflow for direct simulation is as follows in this case. As a first step, a regular grid will be defined for the training and conditioning data. This requires determination of cell-size for the grid along with the extent of the grid. For this we can use the function `get.knn` in the package "FNN", or the standard `dist` function of base **R**, although `get.knn` is notably faster. For each of the samples of the Windarling data the distance to the nearest neighbour is computed and the floor of the median distance is chosen as the cell-size. It is important here to note that the cell-size is dictated by the cell-size of the

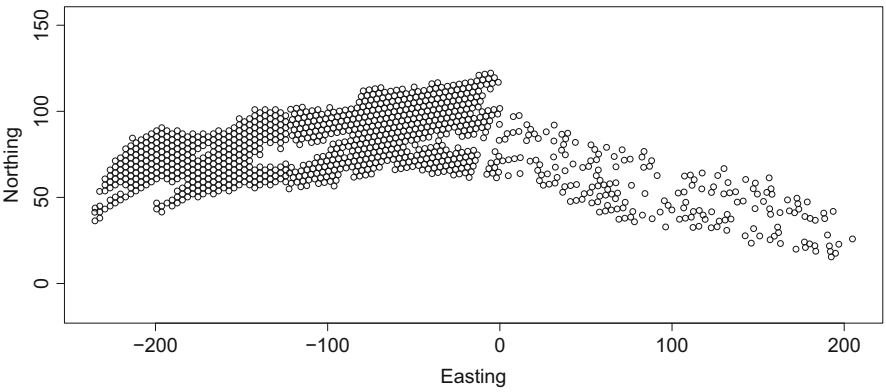

Fig. 10.4 Windarling data sample locations, the western part (Easting < 0 m) will be deemed exhaustive and taken as the training image

training image. It is not possible to generate simulations at a finer resolution.

```
> aux = get.knn(windDS.gcoords, k=1, algo="kd_tree")$nn.dist
> cellSize = floor(median(aux))
> mmin=min(windDS.gcoords[,1])
> mmax=max(windDS.gcoords[,1])
> x = seq(mmin-cellSize/2,mmax+cellSize/2,cellSize)
> mmin=min(windDS.gcoords[,2])
> mmax=max(windDS.gcoords[,2])
> y = seq(mmin-cellSize/2,mmax+cellSize/2,cellSize)
> nx = length(x)
> ny = length(y)
> windDS.ggrid = GridTopology(cellsize=c(cellSize, cellSize),
+   cellcentre.offset=c(min(x), min(y)), cells.dim=c(nx,ny))
> names(windDS.ggrid@cellcentre.offset)=colnames(windDS.gcoords)
```

To check that the sample locations and the grid match, the sample locations and the grid nodes are plotted on the same diagram, as shown in Fig. 10.5. This kind of check should be performed to ensure that the grid is suitable.

```
> plot(windDS.gcoords, asp=1)
> points(coordinates(windDS.ggrid), pch=".", col="red")
```

Next the training data are migrated to the grid by allocating the compositional information at a sample location to the nearest neighbour in the grid.

```
> windDS.gcgrid = windDS.gcoords %>% SpatialPoints %>%
+  SpatialPixelsDataFrame(data=windDS.gcompo, grid=windDS.ggrid)
```

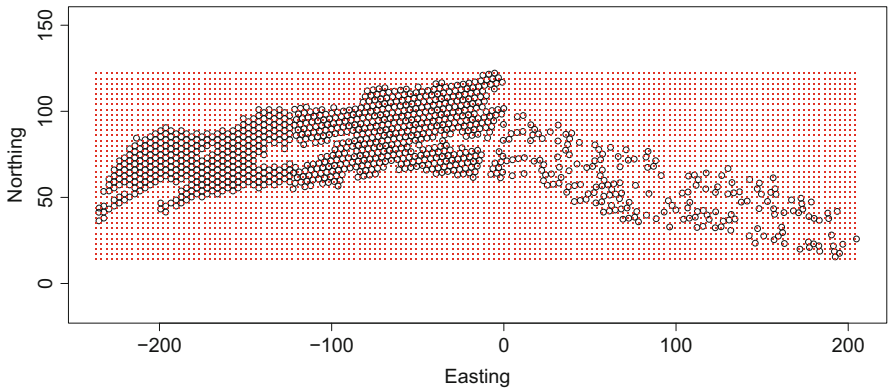

Fig. 10.5 Windarling training and conditioning data locations and underlying regular grid

If you are using a grid generated with `expand.grid`, then the migration can be done using package "KNN"

```
> auxmat = data.frame(matrix(data = NA, nrow=length(x)*length(y),
+                            ncol = ncol(windDS.gcompo)))
> colnames(auxmat) = colnames(windDS.gcompo)
> windDS.gcgrid = cbind(coordinates(windDS.ggrid), auxmat)
> GridMig = get.knnx(coordinates(windDS.ggrid), windDS.gcoords,
+                    k=1, algo="kd_tree")
> windDS.gcgrid[GridMig$nn.index, -(1:2)] = windDS.gcompo
```

One way or another, the migration needs to be checked to ensure that the training images are fit for purpose. This can be done by for example plotting the migrated data and the input training data, as done here for Fe and shown for four of the six components in Fig. 10.6,

```
> myfun = function(j){
+   bks = quantile(windDS.gcompo[,j], prob=(0:10)/10, na.rm=T)
+   out = windDS.gcgrid %>%
+     image_cokriged(ivar=j, legendPos = "top", breaks = bks)
+   return(out)
+ }

> par(mar=c(0.5,5,4,0.1))
> oo = myfun("Fe")

> plot(windDS.gcoords, pch=19, asp=1, cex=0.75,
+      col=oo$col[cut(windDS.gcompo[,"Fe"], breaks=oo$breaks)])
```

or via QQ-plots of the migrated data against the input data, as shown in Fig. 10.7,

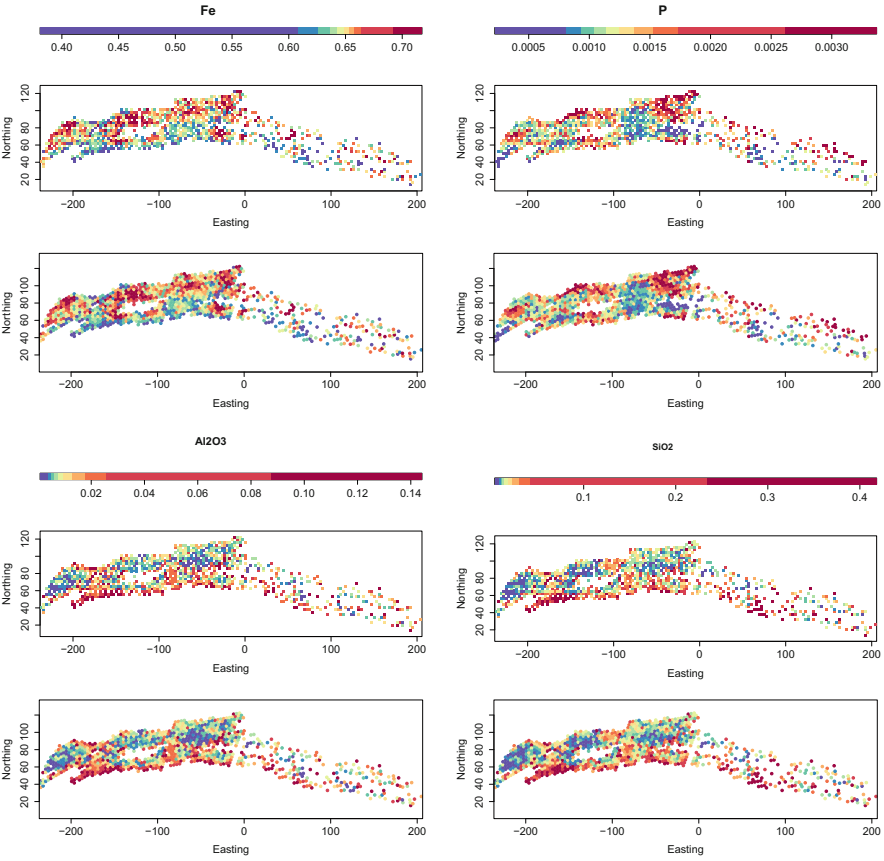

Fig. 10.6 Migrated data (top) and original data (bottom) for Fe, P, Al_2O_3 and SiO_2

```
> par(mfrow=c(2,3))
> for (i in colnames(windDS.gcompo)){
+   qqplot(windDS.gcompo[,i], windDS.gcgrid@data[,i],
+          xlab="true", ylab="migrated", main=i, asp=1)
+   abline(a=0,b=1, col=2)
+ }
```

Having generated the grid and populated it with the compositional information, the next step is to split the data set into the training image and the conditioning data

```
> tk.TI = coordinates(windDS.gcgrid)[,"Easting"]<0
> wind.TI0 = windDS.gcgrid[tk.TI,]
> windDS.cgrid0 =  windDS.gcgrid[!tk.TI,]
```

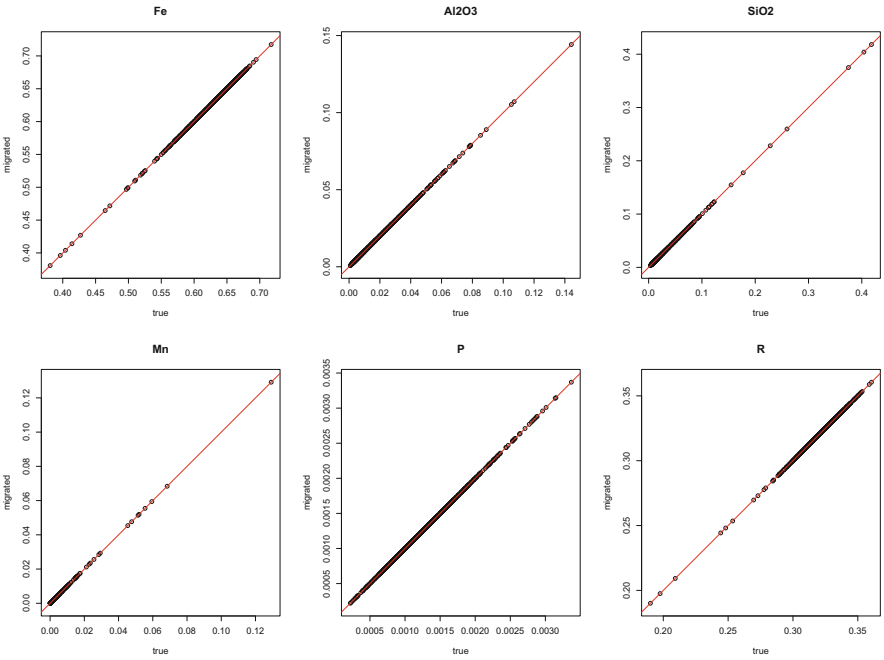

Fig. 10.7 QQ-plots comparing true and migrated data

The grid topology still needs to be restricted to positive easting values, and the training image needs to be reflected in the X direction. We start with the first problem. Remembering that `x` contains the easting coordinates of the centres of the mesh

```
> sum(x>0)

[1] 69

> min(x[x>0])

[1] 0.23
```

we can establish the new topology of the conditioning data (and of the simulation grid!) by editing the relevant properties of the X coordinates from the global grid

```
> windDS.grid = windDS.ggrid
> windDS.grid@cellcentre.offset[1]=min(x[x>0])
> windDS.grid@cells.dim[1] = sum(x>0)
> windDS.grid

                  Easting Northing
cellcentre.offset    0.23       14
cellsize             3.00        3
cells.dim           69.00       37
```

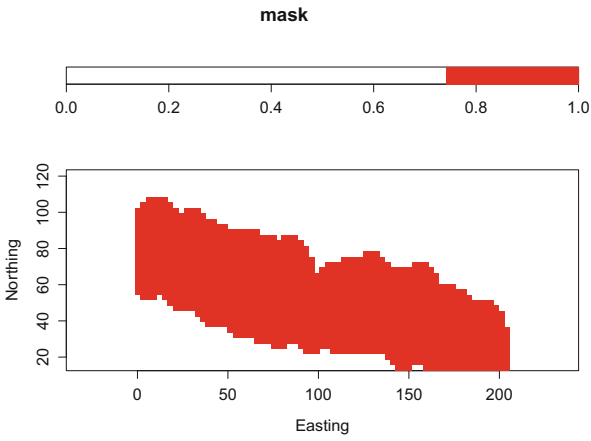

Fig. 10.8 Windarling East simulation region

and now recast the conditioning data to this new common grid, remembering to flip the X coordinates

```
> windDS.cgrid = windDS.cgrid0 %>% as("SpatialPixels") %>%
+   SpatialPixelsDataFrame(windDS.cgrid0@data, grid=windDS.grid)
```

Now the training image is flipped horizontally, directly using command `flipHorizontal` from package "sp" for this goal

```
> wind.TI = flipHorizontal(as(wind.TI0,"SpatialGridDataFrame"))
```

There is also a command `flipVertical`, as such grid manipulations are not uncommon in real applications, where it is necessary that the training image and the simulation grid share spatial scale and possible anisotropy orientations. Lastly a mask is defined to constrain the simulation region. In the first instance, the nodes with conditioning information will be chosen, and then nodes to be simulated will be defined. Here we choose only nodes close to the conditioning data to avoid extreme extrapolations. This is done by defining a maximum distance from the conditioning locations, in this case set to 10 m. The resultant mask is shown in Fig. 10.8.

```
> windDS.mask=constructMask(grid=windDS.grid, method="maxdist",
+                   maxval = 10, x = as.data.frame(windDS.cgrid))

> image(windDS.mask, col=c(NA, "red"))
```

We are now ready to construct the spatial model object and run the simulations,

```
> windDS.cmodel = make.gmCompositionalMPSSpatialModel(
+               data = windDS.cgrid, V = "ilr", model = wind.TI )
```

As before, the parameters need to be set to define the pattern size, the fraction of the training image to be scanned, the threshold t and the number of realisations.

```
> windDS.pars =
+   DSpars(nsim=25, scanFraction=0.75, patternSize=6, gof=0.05)
```

```
> wind.dsim =  predict(windDS.cmodel, pars=windDS.pars,
+                  newdata=setMask(windDS.grid, mask=windDS.mask))
```

Once the simulations have been generated, save them along with the input parameters for future evaluation and postprocessing.

```
> save(windDS.pars, windDS.cmodel, windDS.cgrid, wind.dsim,
+      file="Windarling_TIsim_East.RData")
```

An arbitrarily selected realisation is shown in Fig. 10.9. The realisation appears to reflect the input data well, although this will require further analysis. We should note here that the simulations appear rather noisy, a consequence of the limited size of the training image. The following code would allow exploring all simulations

```
> wind.acomp=acomp(wind.compo)
> myfun = function(i,j){
+   bks = quantile(wind.acomp[,j], prob=0:10/10)
+   out =  getStackElement(wind.dsim,i) %>%
+     image_cokriged(ivar=j, legendPos = "top", breaks=bks)
+ }

> manipulate(myfun(i,j),
+            i=slider(1, max = 25),
+            j=picker("Fe", "Al2O3", "SiO2", "Mn", "P", "R"))
```

Problem 11.1 proposes an evaluation of the results; readers should make sure that they have stored input and output of this simulation as indicated before.

Problems

10.1 Impact of the Scan Fraction on the DS Simulation Results

In the direct sampling simulation based on a sample of the Tellus subcomposition, the scan fraction was set to 0.75. Generate one realisation of the subcomposition for the following scan fractions: 0.3, 0.4, 0.5, 0.6 and 0.7. Leave all other parameters unchanged. Compare the resulting realisations.

10.2 Impact of the Pattern Size on the DS Simulation Results

In the direct sampling simulation based on a sample of the Tellus subcomposition, the pattern size was set to 16. Generate one realisation of the subcomposition for the following pattern sizes: 4, 8, 12 and 20. Leave all other parameters unchanged. Compare the resulting realisations.

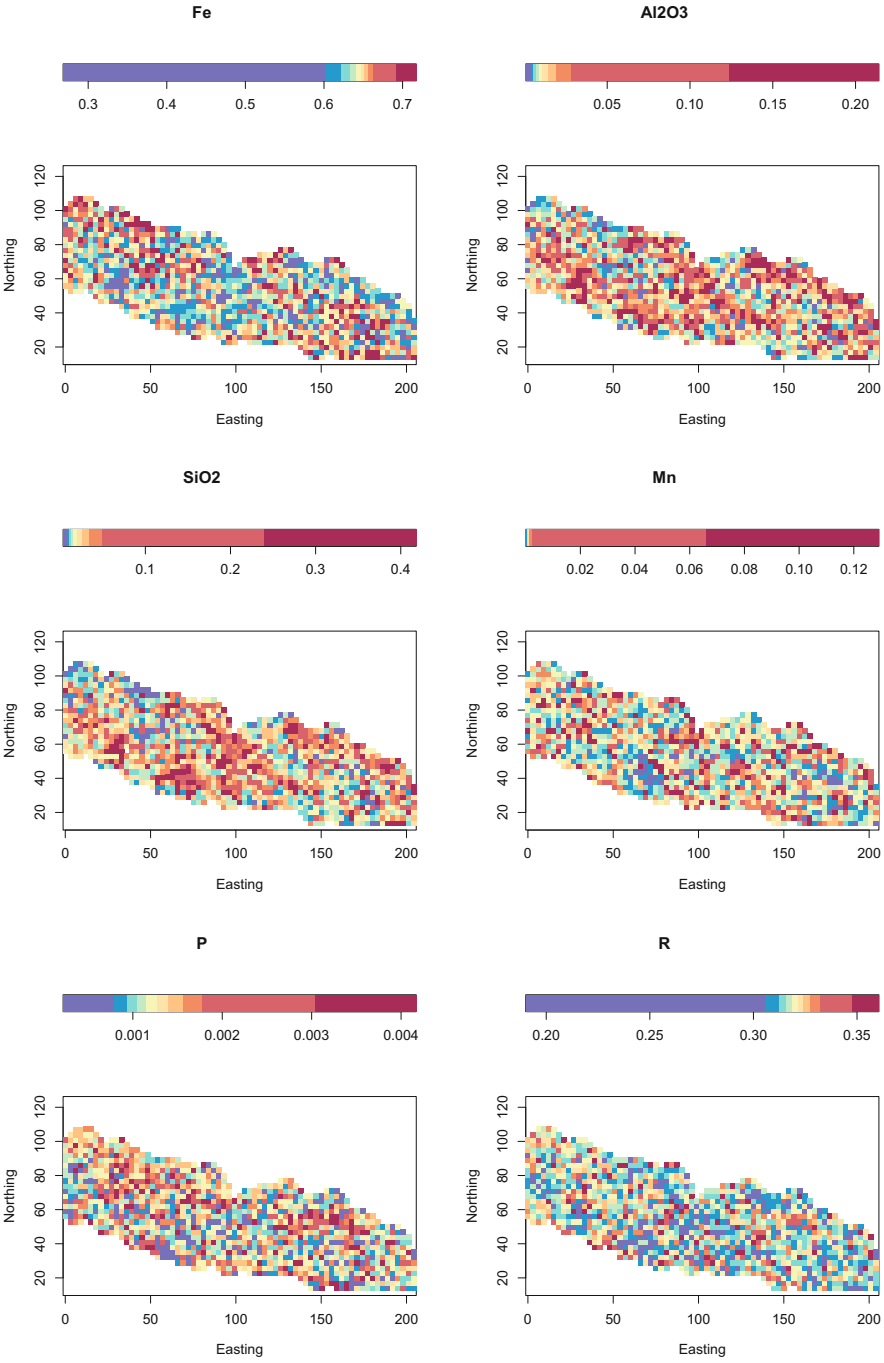

Fig. 10.9 One randomly chosen DS realisation of compositions in the Eastern part of the Windarling bench

References

Bananijar, E., Shargi, Y., & Mariethoz, G. (2019). MPS-APO: a rapid and automatic parameter optimizer for multiple-point geostatistics. *Stoch Environ Res Risk Assess, 33*, 1969–1989.

Boogaart, K. G. v. d., Tolosana-Delgado, R., Lehmann, M., & Mueller, U. (2018). On the joint multipoint simulation of discrete and continuous geometallurgical parameters. In R. Dimitrakopoulos (Ed.), *Advances in orebody modelling and strategic mine planning* (pp. 745–765). Cham: Springer.

Emery, X., & Lantuéjoul, C. (2014). Can a training image be a substitute for a random field model? *Mathematical Geosciences, 46*(2), 133–147.

Mariethoz, G., & Caers, J. (2015). *Multiple-point geostatistics: stochastic modeling with training images* (364 pp.). John Wiley & Sons.

Mariethoz, G., & Renard, P. (2010). The direct sampling method to perform multiple-point geostatistical simulations. *Water Resources Research, 46*(11), W11536.

Meerschman, E., Pirot, G., Mariethoz, G., Straubhaar, J., Meirvenne, M. V., & Renard, P. (2013). A practical guide to performing multiple-point statistical simulations with the direct sampling algorithm. *Computers and Geosciences, 52*, 307–324.

Talebi, H., Mueller, U., & Tolosana-Delgado, R. (2019). Joint simulation of compositional and categorical data via direct sampling technique–application to improve mineral resource confidence. *Computers & Geosciences, 122*, 87–102.

Chapter 11
Evaluation and Postprocessing of Results

Abstract In this chapter tools for the evaluation of results from estimation and simulation are considered. We include QQ-plots, PP-plots, variogram swarms and misclassification analysis to assess how well the output data reflect the inputs. In addition we consider upscaling of simulation results and different approaches for this.

11.1 Evaluation of Results

In this final chapter we consider the evaluation of simulation results. There are certain minimum requirements (Leuangthong et al., 2004) that should be satisfied before accepting the results of a simulation. These include reproduction of key statistics such as means, histograms and target variograms, but also credibility of maps generated from the data.

11.1.1 Validation of the Simulation Model

In Sect. 9.4 we discussed the use of validation approaches based on kriging or cokriging for Gaussian simulation methods. For other methods, the appraisal needs to be based on the simulation output at the sample or validation locations. No function exists in "`gstat`" that allows a simple leave one out validation for simulation, and we need to resort to using the jackknife approach as illustrated in Sect. 7.3.

If a validation set is available, then two possible approaches arise, depending on the nature of the simulation algorithm and the space in which validations are desired. In the case of validating Gaussian scores, again (co-)kriging estimates of these scores at the validation locations can be used to calculate error statistics as illustrated in Chap. 7. The criteria as to whether results are acceptable then are the same as outlined on page 136.

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3_11

If the algorithm to be used is not Gaussian, then an alternative approach is to perform the simulation at the validation data locations and compare the simulated realisations with the true values directly. In this situation, we will have a collection of L cross-validation simulated residuals $\left\{\epsilon^{\ell}(\mathbf{x}_\alpha) = z(\mathbf{x}_\alpha) - z^{\ell}(\mathbf{x}_\alpha) : \ell = 1, \ldots, L; \alpha = 1, \ldots, J\right\}$ for each of the J locations in the jackknife set for the univariate case; or in the multivariate case $\left\{\boldsymbol{\epsilon}^{\ell}(\mathbf{x}_\alpha) = \mathbf{z}(\mathbf{x}_\alpha) - \mathbf{z}^{\ell}(\mathbf{x}_\alpha) : \ell = 1, \ldots, L; \alpha = 1, \ldots, J\right\}$. With them:

1. The mean error, *ME*, can be computed as, respectively,

$$ME = \frac{1}{JL}\sum_{\alpha=1}^{J}\sum_{\ell=1}^{L}\epsilon^{\ell}(\mathbf{x}_\alpha), \quad \mathbf{ME} = \frac{1}{JL}\sum_{\alpha=1}^{J}\sum_{\ell=1}^{L}\boldsymbol{\epsilon}^{\ell}(\mathbf{x}_\alpha).$$

2. The mean squared error, *MSE*, is, respectively,

$$MSE = \frac{1}{JL}\sum_{\alpha=1}^{J}\sum_{\ell=1}^{L}(\epsilon^{\ell}(\mathbf{x}_\alpha))^2, \quad MSE = \frac{1}{JL}\sum_{\alpha=1}^{J}\sum_{\ell=1}^{L}\|\boldsymbol{\epsilon}^{\ell}(\mathbf{x}_\alpha)\|^2.$$

3. In general, none of the mean squared deviation ratios for the univariate or multivariate cases can be computed from simulations.

The first two error measures can be computed with the command `xvErrorMeasures`.

For a simulation, accuracy and the associated measures precision and goodness can be computed from the residuals calculated for the simulated realisations at the validation locations by subtracting the true value from the simulated value. In the univariate case, the accuracy plot described in Sect. 7.4 is defined as follows. For each validation location $\mathbf{x}_\alpha$ arrange the simulation residuals $\left\{\epsilon^{\ell}(\mathbf{x}_\alpha) : \ell = 1, \ldots, L\right\}$ in increasing order, denoted by $\left\{\epsilon^{(\ell)}(\mathbf{x}_\alpha)\right\}$. For each $\mathbf{x}_\alpha$ and p, $0 \leq p \leq 1$, we may then define an indicator variable as

$$i(\mathbf{x}_\alpha, p) = \begin{cases} 1 \text{ if } \quad \epsilon^{(\text{floor}(L(1-p)/2))}(\mathbf{x}_\alpha) \leq 0 \leq \epsilon^{(\text{ceiling}(L(1+p)/2))}(\mathbf{x}_\alpha) \\ 0 \text{ otherwise} \end{cases}.$$

This definition enables calculation of the coverage of each interval with Eq. (7.5). The subsequent calculations for accuracy, precision and goodness can then be used unmodified as defined in Sect. 7.4.

The function `accuracy` performs these calculations for simulations. The arguments of the function depend on the class of the container `x` of the realisations. If this is an object of class "`data.frame`", then the column names of the realisations must contain the string `"sim"` (e.g. `"sim1"`, `"sim2"`, etc), and the extra argument `method = "simulation"` must be included:

```
> a = accuracy(x, observed, method = "simulation")
```

If the realisations are contained in an object of class "`DataFrameStack`", then the syntax is

```
> a = accuracy(x, observed, vars)
```

where `vars` must name the variable to be tested. Both `x` and `observed` *must* have the variable `observed` in its column names, and `x` in its non-stacking dimension (i.e. the dimension gathering the variables). The same syntax can be used for realisations from multivariate simulation, although this requires an additional argument `method`. The reason for the additional argument is the well-known fact that, in general, there is no natural order relation for multivariate data. We therefore establish an ordering based on a scalar norm. Two methods presently exist: `method="Mahalanobis"` and `method="flow"`. For `method="Mahalanobis"`, it is assumed that the realisations `x` and the true values `observed` are provided in the appropriate representation (i.e.: in alr or ilr scores) and that the Mahalanobis geometry makes sense in that representation. Otherwise, `method="flow"` enables one to perform the same calculations in multivariate Gaussian space, after first transforming the realisations and the true value through a flow anamorphosis (as explained in Chap. 8). This is done independently for each location $\mathbf{x}_\alpha$.

For either method the calculation proceeds as follows: At each validation location $\mathbf{x}_\alpha$ the mean vector $\bar{\mathbf{z}}(\mathbf{x}_\alpha)$ and covariance matrix $\hat{\boldsymbol{\Sigma}}(\mathbf{x}_\alpha)$ are computed based on the realisations $\{\mathbf{z}^\ell(\mathbf{x}_\alpha); i = 1, \ldots, L\}$. For each location $\mathbf{x}_\alpha$ and each ℓ, $1 \leq \ell \leq L$ the following square distances are computed: the square Mahalanobis norm of the true value

$$u_0(\mathbf{x}_\alpha) = d_M^2(\mathbf{z}(\mathbf{x}_\alpha), \bar{\mathbf{z}}(\mathbf{x}_\alpha) | \hat{\boldsymbol{\Sigma}}(\mathbf{x}_\alpha))$$

and the square Mahalanobis norm

$$u_{(\ell)}(\mathbf{x}_\alpha) = d_M^2(\mathbf{z}^\ell(\mathbf{x}_\alpha), \bar{\mathbf{z}}(\mathbf{x}_\alpha) | \hat{\boldsymbol{\Sigma}}(\mathbf{x}_\alpha)).$$

The square norms $\{u_{(1)}(\mathbf{x}_\alpha), u_{(2)}(\mathbf{x}_\alpha), \ldots, u_{(L)}(\mathbf{x}_\alpha)\}$ are arranged in ascending order and the indicator $i(\mathbf{x}_\alpha, p)$ at location $\mathbf{x}_\alpha$ and probability p is defined as

$$i(\mathbf{x}_\alpha, p) = \begin{cases} 1 \text{ if } & u_0(\mathbf{x}_\alpha) \leq u_{(ceiling(pL))} \\ 0 \text{ otherwise} & \end{cases}.$$

As in the univariate case we can calculate the coverage of each interval with Eq. (7.5) and follow the calculations in Sect. 7.4 for obtaining accuracy, precision and goodness.

11.1.2 Individual Realisation Maps

A first step in any evaluation of results is a visualisation of several realisations to check for artefacts and match with the sample data. To this end the sample data can be superimposed on the spatial map of the realisation under consideration. For example, to check the cosimulation results of the Windarling data generated in Chap. 9, the function `manipulate` can be used, as shown in the sample code on page 175.

11.1.3 Statistical Maps

In addition to maps of realisations one can compute a map of the so-called E-Type mean over the realisations, which is nothing other but a map of location-wise averages (Goovaerts, 1997; Rossi & Deutsch, 2014). In the case of compositional data the *E-type mean* is computed as the closed geometric mean over the realisations, in analogy with Eq. (3.1)

$$\mathbf{z}^*(\mathbf{x}) = \mathscr{C}\left[\left(\prod_{\ell=1}^{L} z_1^\ell(\mathbf{x})\right)^{1/L}, \left(\prod_{\ell=1}^{L} z_2^\ell(\mathbf{x})\right)^{1/L}, \ldots \left(\prod_{\ell=1}^{L} z_D^\ell(\mathbf{x})\right)^{1/L}\right]. \tag{11.1}$$

The *total dispersion* of simulated compositions at location $\mathbf{x}$ is given by

$$\text{totvar}(\mathbf{x}) = \frac{1}{2D^2}\sum_{i=1}^{D}\sum_{j=1}^{D}\frac{1}{L-1}\sum_{\ell=1}^{L}\left(\ln\left(\frac{z_i^\ell(\mathbf{x})}{z_j^\ell(\mathbf{x})}\right) - \ln\left(\frac{z_i^*(\mathbf{x})}{z_j^*(\mathbf{x})}\right)\right)^2. \tag{11.2}$$

At a node in the grid that corresponds to a conditioning node, the E-Type mean is equal to the conditioning composition and the global dispersion is 0. For example, the E-Type mean and total dispersion for the realisations derived from the sequential cosimulation of the Windarling sample data can be computed as follows:

```
> wind.cosim.Etype = wind.compo.cosim %>%
+   gmApply(c("loc","var"), geometricmean) %>% clo
> wind.sim.Etype = wind.compo.sim %>%
+   gmApply(c("loc","var"), geometricmean) %>% clo
> wind.cosim.totvar = wind.compo.cosim %>%
+   gmApply("loc", function(x) mvar(acomp(x)) )
```

```
> aux = cbind(wind.grid.fine, wind.cosim.Etype)
> myfun = function(j){
+   out = image_cokriged(aux, ivar=j, legendPropSpace = 0.2,
+                        legendPos = "top")
+ #  points(wind.coords, pch=21, col=1, cex=0.5,
+ #         bg=out$col[cut(wind.acomp[,j],out$breaks)])
+ }
```

```
> sapply(c("Fe", "Al2O3", "SiO2", "Mn", "P", "R"), myfun)
```

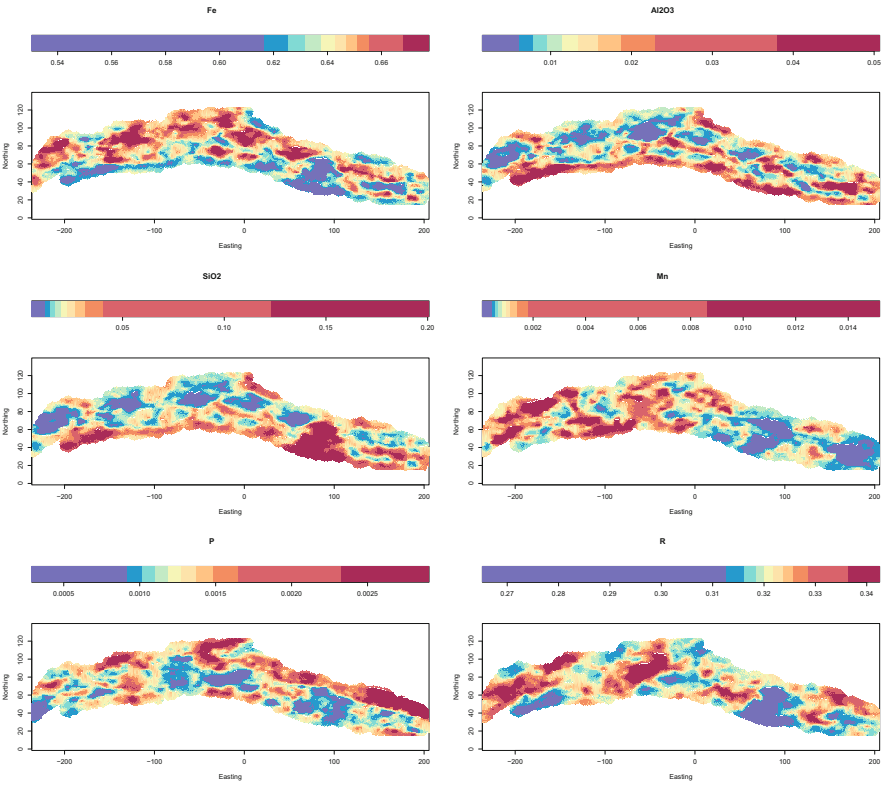

Fig. 11.1 E-type estimates of the expected value of the composition in raw scale for the Windarling data set, based on sequential Gaussian simulations

Spatial maps of the E-type means derived from the simulation of the Windarling data obtained with techniques from Chap. 9 are shown in Fig. 11.1. Figure 11.2 shows the associated global dispersion that is high at the boundaries and low near sample locations.

```
> aux = cbind(wind.grid.fine, totvar=wind.cosim.totvar)
> image_cokriged(aux, ivar="totvar", legendPos = "top")
> #points(wind.coords, pch=21, col=1, cex=0.25)
```

In addition to maps summarising the mean and total dispersion, maps can be produced for example reflecting threshold values of interest for specific components, which might be of interest for highlighting hotspots of the spatial distribution. These maps are derived from the local distributions of the realisations. Given a threshold t, we can determine the frequency of exceeding the threshold at each location and also compute the average above the chosen threshold. For instance, the following allows computing the probability that the ratio of (Mn+P)/Fe exceeds 0.4%

```
> names(dimnames(wind.compo.cosim))
```

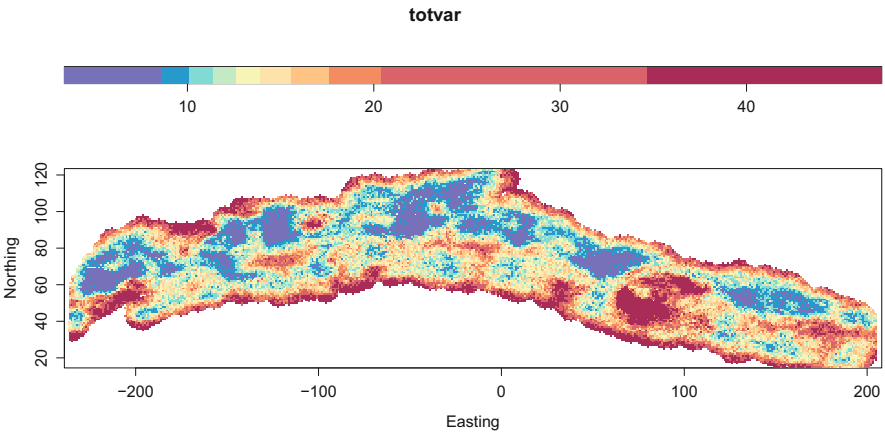

Fig. 11.2 Total variation estimate of the composition for the Windarling data set, based on sequential Gaussian simulations

```
[1] "loc" "var" "sim"
```

```
> myfun = function(x) mean(((x["Mn",]+x["P",])/x["Fe",])>0.004)
> aux = gmApply(wind.compo.cosim, MARGIN="loc", FUN= myfun)
> aux = cbind(wind.grid.fine, risk=aux)
> out = image_cokriged(aux, ivar="risk", legendPos = "top")
> del_ratio = (wind.compo$Mn+wind.compo$P)/wind.compo$Fe
> points(wind.coords, pch=22, cex=0.5, col=1,
+        bg=out$col[c(1,10)][1+(del_ratio>0.004)])
```

Note that these calculations may be quite tricky to do, in particular when defining the ad hoc function `myfun` for the operation desired. In this case, the stacked data can be seen in analogy with an array with dimensions along locations, variables and simulations, in this order (as shown by the first row of the chunk, using `dimnames`). As a consequence, `gmApply(..., MARGIN='loc', FUN=myfun)` will make `myfun` operation a matrix of dimensions along variables and simulations, and to extract Fe or Mn from it we need to compute `x["Fe",]` resp. `x["Mn",]`. On the other hand, in the original data, the components are placed in columns, as usual, and can be recovered with the `$` syntax.

11.1.4 Reproduction of Target Marginal and Two-Point Statistics

In addition to spatial maps any evaluation needs to encompass reproduction of first and second order moments. Thus for each realisation geometric means are needed, as well as histograms, variograms and also ternary plots. Histograms (or kernel density estimates) and variograms are of relevance irrespective of the nature of the

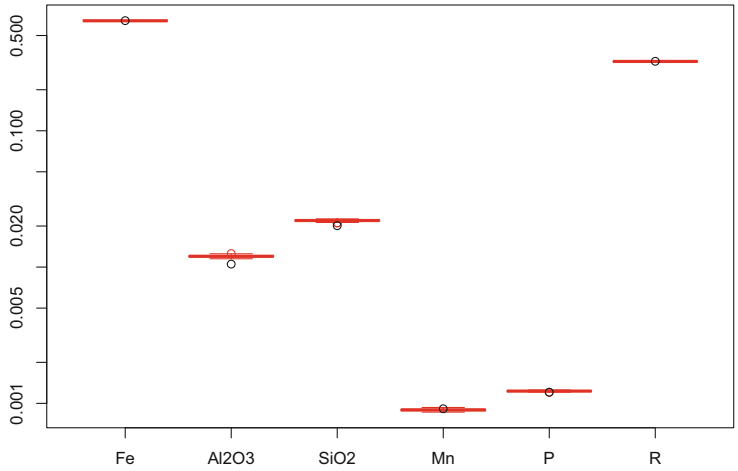

Fig. 11.3 Boxplots of the simulated compositional means of the whole Windarling deposit, compared with the compositional means of the data (circles)

data, while ternary plots capture compositional aspects of the realisations. Functions `gmApply` and `swarmPlot` can be used.

This chunk shows how to compute compositional means of the realisations and compare them with the compositional mean value of the data

```
> a =  clo(gmApply(wind.compo.cosim, MARGIN=c("sim","var"),
+                  FUN= geometricmean, na.rm=T))
> boxplot(data.frame(a), log="y", border=2)
> points(1:6, mean(wind.acomp), lty=2)
```

Results are shown in Fig. 11.3.

To check histogram reproduction, QQ-plots are suitable, which display the quantiles of the sample distribution against those of the realisation as introduced in Fig. 3.9 of Chap. 3 for comparing quantiles of pairwise log-ratios against the normal distribution. In particular, swarms of QQ-plots allow a quick appraisal of the histogram reproduction for the entire set of realisations. The function `swarmPlot` is useful for that. To use it, we need to write an ad hoc function that takes one realisation of the simulation stack and returns a list or a data frame with two elements: the `x` and the `y` values of the plot:

```
> myQQfun = function(x, j="Fe"){
+   erg = qqplot(wind.acomp[,j], x[,j], plot.it = F)
+   return(data.frame(erg))
+ }
> a = myQQfun(getStackElement(wind.compo.cosim.aux,1))
> summary(a)

       x                y
 Min.   :0.268   Min.   :0.351
 1st Qu.:0.623   1st Qu.:0.619
```

```
 Median :0.644   Median :0.641
 Mean   :0.633   Mean   :0.632
 3rd Qu.:0.657   3rd Qu.:0.654
 Max.   :0.717   Max.   :0.694
```

Once the ad hoc function is defined, `swarmPlot` can be used

```
> auxFe = swarmPlot(wind.compo.cosim.aux, PLOTFUN = myQQfun,
+   j="Fe", .plotargs = list(asp=1, xlab="observed quantiles",
+                  ylab="simulation quantiles", main="Fe") )
> abline(a=0, b=1, col=2)
```

to produce the swarms as for instance in the upper left plot of Fig. 11.4. The diagrams of this figure show the QQ-plots for each of the six variables of Windarling: for each realisation, a grey line is displayed. The red reference line (the identity curve) shows the ideal behaviour of a perfect reproduction of the distribution of the data. The function `swarmPlot` admits a list of arguments for controlling the appearance of the plot in the argument `.plotargs`. This list can have as named elements any of the plotting arguments of functions `plot` or `plot.default`. Several chunks of this section illustrate how to use this argument. Alternatively, you can give `.plotargs=F`, which suppresses plotting.

If numerical summaries are required, one measure of use is the distribution mean square deviation, computed as the mean square difference between the quantiles of the input data and those of the simulated realisations. These can then be visualised as boxplots to check for "atypical" realisations. To compute them, again the functions `swarmPlot` and `sapply` will be useful: `swarmPlot` returns the results of its calculations as an invisible object of class "`swarmPlot`" that can be replotted (with the `plot.swarmPlot` method) or used for further calculations, e.g. for producing the distributional divergence mentioned above

```
> meanSqDev = function(x) mean((x$x-x$y)^2)
> statHistFe = sapply(auxFe, meanSqDev)
> boxplot(statHistFe)
```

Figure 11.5 shows the boxplots of these divergences for each one of the variables: one can easily see that the distribution of P is excellently reproduced, while Fe and SiO2 are notably worse. Other measures of deviation between observed and simulated distribution can be used, for instance the Hellinger divergence

$$H(f_X(x), g_X(x)) = \int (f_X^{1/2}(x) - g_X^{1/2}(x))^2 dx,$$

which would require the definition of the function

```
> HellingerDev = function(x)
+   mean((diff(x$x)^0.5-diff(x$y)^0.5)^2)
```

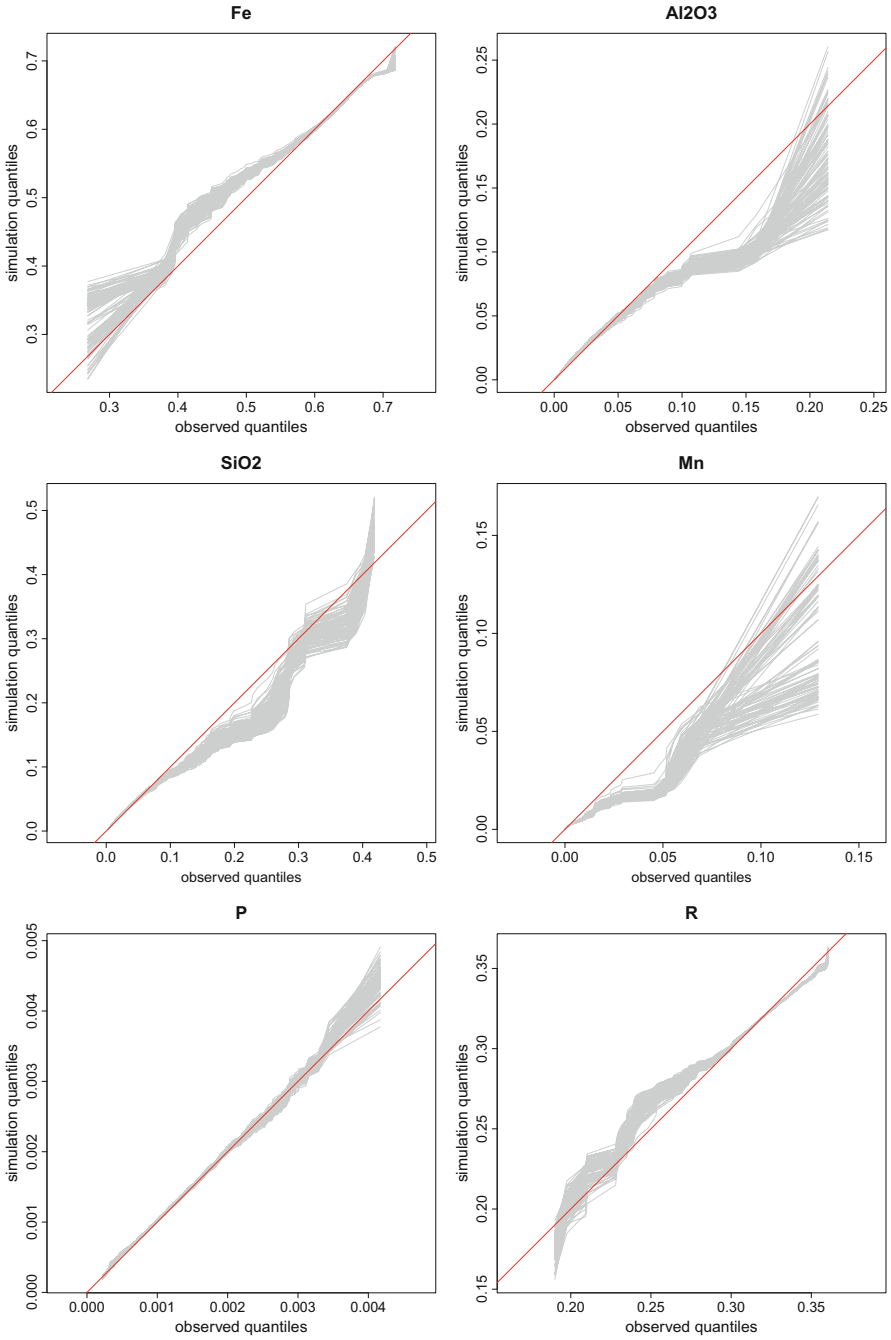

Fig. 11.4 Swarm QQ-plots of the realisations of the composition at Windarling, compared with the observed distribution of each variable

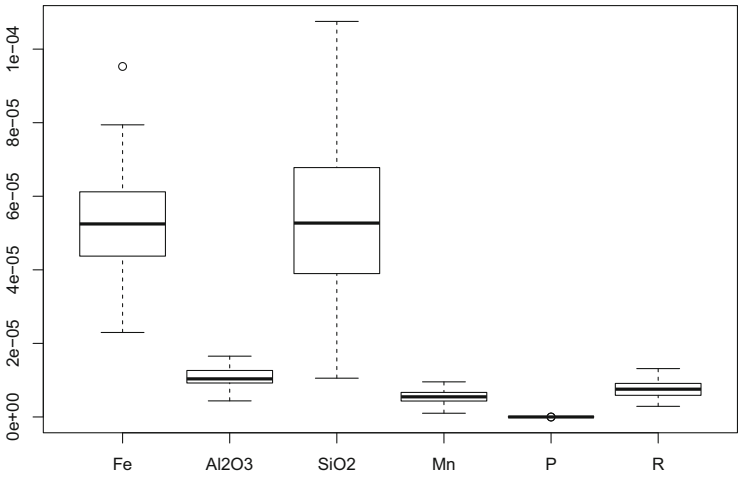

Fig. 11.5 Boxplots of the distribution mean square deviations for each simulation and each variable

To check two-point statistics apart from computing the dispersion over each realisation, variogram reproduction can be checked by comparing the experimental semivariograms of the realisations with the variograms of the conditioning data (Leuangthong et al., 2004). To do so the experimental direct and cross variogram can be plotted together with those of the input data:

```
> vgFe =
+   gstat(id="Fe", formula=Fe~1, locations=~Easting+Northing,
+         data = cbind(data.frame(wind.acomp), wind.coords)) %>%
+   variogram(cutoff=50)
> myVGfun = function(x){
+ vg = gstat(id="Fe", formula=Fe~1, locations=~Easting+Northing,
+         data = cbind(data.frame(x), wind.grid.fine.masked)) %>%
+                 variogram(cutoff=50)
+   return(data.frame(x=c(0,vg$dist),y=c(0,vg$gamma)))
+ }
> aux = swarmPlot(wind.compo.cosim.aux, PLOTFUN= myVGfun,
+                 .plotargs=F)
> plot(aux, xlim=range(0, vgFe$dist), ylim=range(0, vgFe$gamma),
+      xlab="lag distance", ylab="semivariogram")
> points(vgFe$dist, vgFe$gamma, col=2)
```

Here one would like to see the target variograms lie within the swarm of realisation variograms, particularly if the comparison is based on variograms of normal scores. However, the smoothing during the various back-transformations might result in under-estimation of the variability and hence in lower variogram sills. The overall shape of the variogram should be preserved, though. For the Windarling cosimulation this is the case for Fe, while P shows an excellent reproduction (Fig. 11.6). Other variograms were not shown to save space. Note that variogram

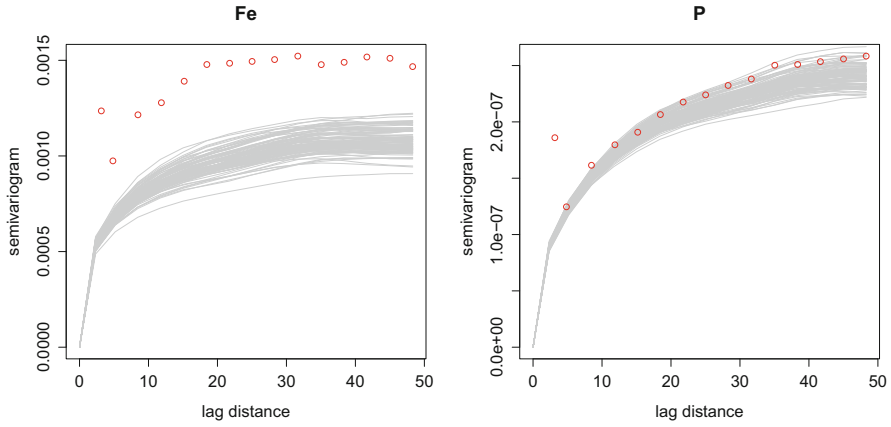

Fig. 11.6 Variogram swarms for Windarling Fe and P simulated realisations, compared with the variograms of the original data (red dots)

reproduction can also be tested for multipoint simulations, if the model captured by the training image is sufficiently stationary.

11.1.5 Selectivity Curves

Beyond the classical statistical descriptions (means, variances, variograms, QQ), in mining and in environmental monitoring it is also common to use two further summaries of the global conditional distribution useful for economic assessments: the tonnage and the grade above cutoff curves (David, 1977; Lantuéjoul, 1990; Abzalov, 2016). Both curves are only defined in the univariate context but can be used for understanding the marginal distribution of one of the variables in a compositional setting. Both curves are computed from the realisations $\{z^{\ell}(\mathbf{x}_{\alpha}); \ell = 1, 2, \ldots, L\}$ at each of the M locations $\mathbf{x}_{\alpha}$ and a sequence of K reference levels or *cutoff values* $c_1, c_2, \ldots c_K$. At a cutoff level c, the *tonnage above cutoff* reports the proportion of volume with $Z(\mathbf{x}) > c$

$$T(c) = \int I_{Z>c}(\mathbf{x}) d\mathbf{x}, \qquad I_{Z>c}(\mathbf{x}) = \begin{cases} 1 \text{ if } \quad Z(\mathbf{x}) > c \\ 0 \text{ otherwise} \end{cases}.$$

The *grade above cutoff* is then defined as the average value of $Z(\mathbf{x})$ for those locations where the grade is above the cutoff on the selected volume,

$$G(c) = \frac{1}{T(c)} \int Z(\mathbf{x}) I_{Z>c}(\mathbf{x}) d\mathbf{x},$$

and it makes only sense for an *extensive variable*, i.e. an additive variable.

These values can be estimated for each realisation ℓ and each reference cutoff c_k as the proportion of grid locations $\mathbf{x}_\alpha$ with $z^\ell(\mathbf{x}_\alpha) > c_k$,

$$\hat{T}^\ell(c_k) = \frac{1}{M}\sum_{\alpha=1}^{M} i(z^\ell(\mathbf{x}_\alpha), c_k) \qquad i(z^\ell(\mathbf{x}_\alpha), c_k) = \begin{cases} 1 \text{ if } & z^\ell(\mathbf{x}_\alpha) > c_k \\ 0 \text{ otherwise} & \end{cases},$$

respectively

$$\hat{G}^\ell(c_k) = \frac{1}{M\hat{T}^\ell(c_k)}\sum_{\alpha=1}^{M} z^\ell(\mathbf{x}_\alpha) i(z^\ell(\mathbf{x}_\alpha), c_k).$$

A grade above cutoff curve is then a plot of $G(c_k)$ against the sequence of cutoffs $c_k, k = 1, \ldots, K$ and the tonnage above cutoff curve that of $T(c_k)$ against $c_k, k = 1, \ldots, K$. Swarm plots of these curves inform us of the actual effect of uncertainty on issues such as the economic value of a mineral deposit, the costs of a mining or a remediation operation, or the level of exposure of an ecosystem under a pollution episode. Additionally, these curves can also be estimated for the data, and compared to the swarm: important discrepancies will point to possible problems of the modelling process.

No closed form function is provided for such calculations, so we will do them here step by step. The first part will consist on establishing two functions that do the calculations for a set of values of z and cutoffs c

```
> TonnageCurve = function(X, vr=1, cutoffs=
+       seq(from=min(X[,vr]), to=max(X[,vr]), length.out=201)){
+   t = rowMeans(outer(cutoffs, X[,vr], "<="))
+   return(data.frame(cutoff=cutoffs,tonnage=t))
+ }
> GradeCurve = function(X, vr=1, cutoffs=
+       seq(from=min(X[,vr]), to=max(X[,vr]), length.out=201)){
+   z = c(X[,vr])
+   t = rowMeans(outer(cutoffs, z, "<="))
+   g = outer(cutoffs, z, "<=") %*% z
+   return(data.frame(cutoff=cutoffs,grade=g/(t*length(z))))
+ }

> par(mfrow=c(1,2))
> TonnageCurve(wind.acomp, vr="Fe") %>% plot(type="l")
> GradeCurve(wind.acomp, vr="Fe") %>% plot(type="l")
```

The resulting plots are not included to save space. Once we have these functions, we can pass them with the data frame stack to `swarmPlot`

```
> # tonnage:
> aux = swarmPlot(wind.compo.cosim.aux, PLOTFUN=TonnageCurve,
+                 vr="Fe", .plotargs = F)
> plot(aux, xlab="cutoff", ylab="tonnage", main="Fe")
> aux0 = TonnageCurve(wind.acomp, vr="Fe")
> lines(aux0[,1], aux0[,2], col=2)
```

```
> # grade
> aux = swarmPlot(wind.compo.cosim.aux, PLOTFUN=GradeCurve,
+                 vr="Fe", .plotargs = F)
> plot(aux, xlab="cutoff", ylab="grade", main="Fe")
> aux0 = GradeCurve(wind.acomp, vr="Fe")
> lines(aux0[,1], aux0[,2], col=2)
```

Figure 11.7 shows the grade and tonnage curves for each variable of Windarling, independently of each other. The swarms show good reproduction of the curves for the data. There is evidence of grade uncertainty for variables such as Al_2O_3 or Mn at higher cutoffs, whereas the other variables are much more constrained.

11.2 Postprocessing

11.2.1 Block-COK Through Local Simulation

In the preceding sections we have considered geostatistical methods for compositional data at *point support*. This is adequate for mapping exercises, but in many applications estimates over larger areas or volumes are required, for example in mining, where equipment constraints dictate the smallest volume, the so-called *selective mining unit (smu)*, that can reasonably be extracted. Luckily the preceding methods can be applied to block supports V too. The conventional multigaussianity assumptions underlying linear block cokriging provide an estimate of the average log-ratio within a block, denoted by $\mathbf{Y}_0(V)$. That would be obtained with any block cokriging algorithm. However, in mining applications one requires the conditional distribution of the *mass* $\mathbf{m}(V) = \int_{x\in V} \text{agl}(\mathbf{Y}(x))dx$ of each component $\mathbf{m}(V)$ in a block V. In general, the mass is not given by the volumetric scaling of the average log-ratios:

$$\mathbf{m}(V) = \frac{|V|}{|V|}\int_{x\in V} \text{agl}\left[\mathbf{Y}(x)\right]dx \neq |V|\cdot\text{agl}\left[\frac{1}{|V|}\int_{x\in V}\mathbf{Y}(x)dx\right] = |V|\cdot\text{agl}\left[\mathbf{Y}_0(V)\right].$$

One method for estimating the target quantity $\mathbf{m}(V)$ is to use Monte Carlo simulation on a dense grid $\{x_1,\ldots,x_B\} \in V$ within the target block. For each realisation $\{\mathbf{Y}^{(s)}(x_1),\ldots,\mathbf{Y}^{(s)}(x_B)\}$ the mass $\mathbf{m}^{(s)}(V)$ can be approximated by

$$\mathbf{m}^{(s)}(V) \simeq \sum_j \text{agl}(\mathbf{Y}^{(s)}(x_j))\cdot|V|/B,$$

i.e. by summing the agl-transformed simulated values within the block. If a family of S realisations is available, then one obtains a set of values of $\{\mathbf{m}^{(s)}(V) : s = 1,\ldots,S\}$, which can be used to estimate its expectation,

$$\text{E}[\mathbf{m}(V)] \approx \frac{1}{S}\sum_{s=1}^{S}\mathbf{m}^{(s)}(V) \approx \frac{|V|}{S\cdot B}\sum_{s=1}^{S}\sum_{j=1}^{B}\text{agl}(\mathbf{Y}^{(s)}(x_j)).$$

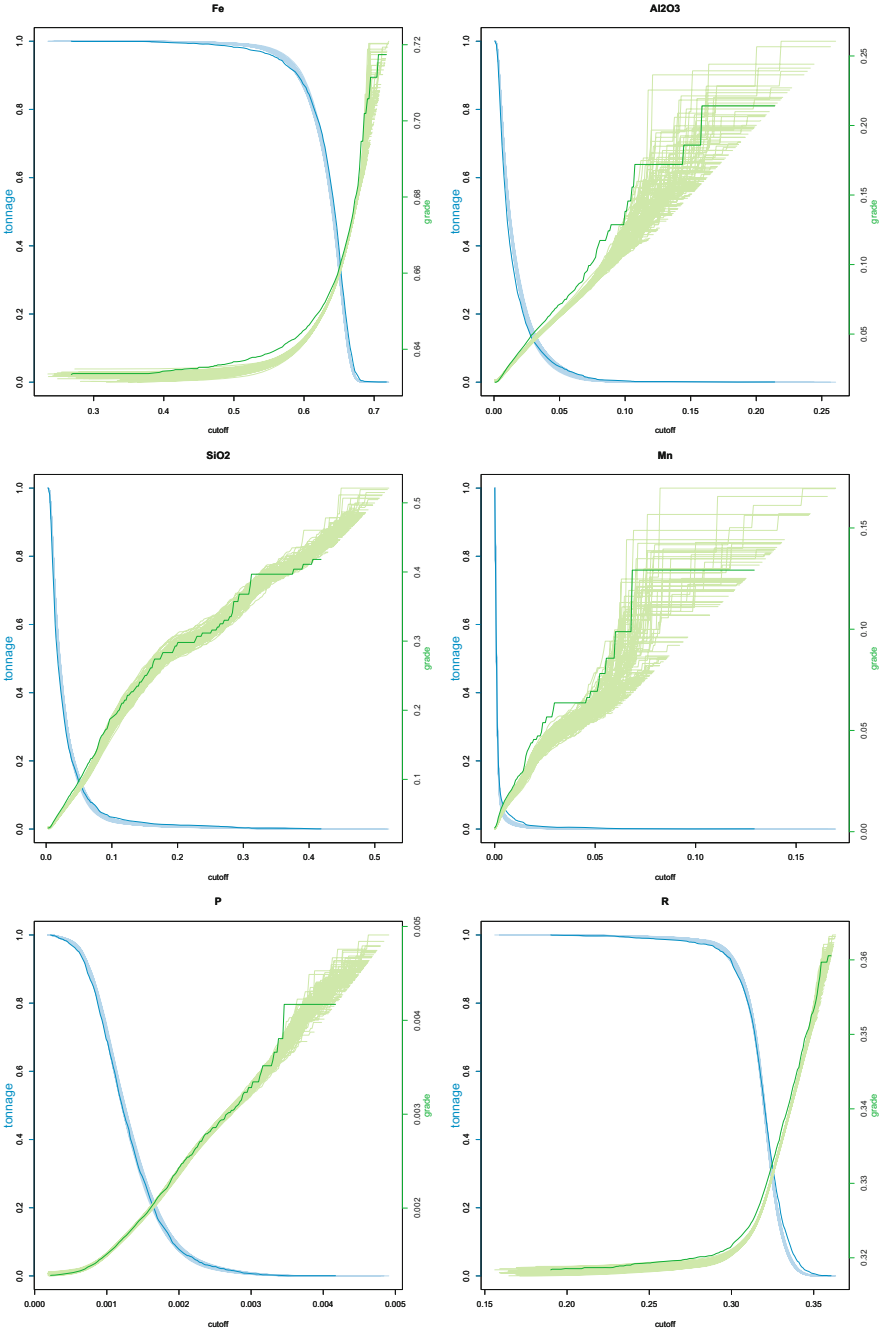

Fig. 11.7 Grade-tonnage curves of each variable at Windarling, based on the cosimulation strategy

While we will be applying sGs here, any other geostatistical simulation procedure could serve, such as LU decomposition that would be particularly suitable, since the number of simulation points is relatively small, as it must only be representative of the variability within the block, not of the regional variability; and with LU one can obtain a large number of simulations at once. As a result a very good approximation of E[$\mathbf{m}(V)$] and even of the entire distribution of $\mathbf{m}(V)$ can be generated.

The easiest way to obtain a 2D-reblocking in **R** is by means of the areal aggregation functions of package "sp" (Sect. 4.1). This requires first packing the data to aggregate and their coordinates into a "SpatialPointsDataFrame" and creating a coarser grid with the blocks for upscaling. For instance here

```
> spMean = SpatialPointsDataFrame(coords=wind.grid.fine,
+                                 data=data.frame(wind.sim.Etype))
> (rgs = gmApply(wind.grid.fine,2,range))

     Easting Northing
[1,]    -236       15
[2,]     205      123

> apply(rgs,2,diff)/5

 Easting Northing
    88.2     21.6

> wind.blockgrid.topo = GridTopology( cells.dim = c(89, 22),
+     cellcentre.offset = c(-240,125), cellsize = c(5,-5)  )
```

we create the "SpatialPointsDataFrame" object bundling the coordinates from wind.grid.fine and as data taking the E-type estimates obtained from the MAF based simulation approach. Then the grid topology is specified to have 89×22 blocks of size $5 \times 5u^2$, with the upper left cell centre so defined that the whole initial data are contained in the coarser grid. Note that the Y grid direction must be specified from the largest to the smallest value; hence with a negative cell-size: this is so because the reference point of "SpatialGrid" topologies is the upper left corner. With these, we just need to aggregate by means of the function mean

```
> wind.blockgrid = SpatialGrid(wind.blockgrid.topo)
> spAggr = aggregate(spMean, by=wind.blockgrid, FUN=mean)
```

Note the necessary conversion of the grid topology to a true "SpatialGrid" here. The result can be plotted either with the functions provided by package "sp"

```
> spplot(spAggr)
```

or if a higher level of control is desired, with any of the plotting facilities provided in preceding chapters, after extracting coordinates and data from the spatial aggregates as shown here (Fig. 11.8)

```
> cbind(coordinates(spAggr), spAggr@data) %>%
+     image_cokriged(ivar="Fe")
```

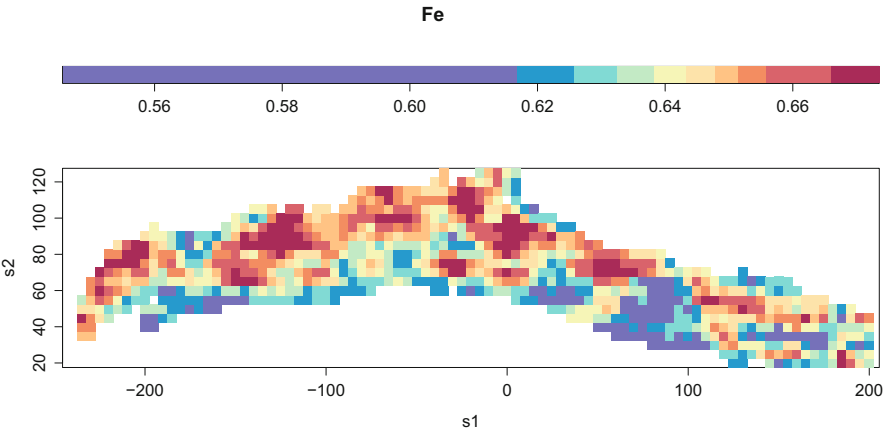

Fig. 11.8 Map of block E-type spatial mean aggregates of Fe

Now that we know how to reblock the E-type estimate, we can do the same for each individual realisation. For this, we need as usual to construct an ad hoc function that does the work for a single data set and then apply that function to the (co)simulation stack

```
> myAggrFun = function(x){
+   spx = SpatialPointsDataFrame(coords=wind.grid.fine,
+                                data=data.frame(x))
+   spAggr = aggregate(spx, by=wind.blockgrid, FUN=mean)
+   return(spAggr@data)
+ }
> wind.compo.sim.block = gmApply(wind.compo.sim, FUN=myAggrFun)
> dim(wind.compo.sim.block)

[1] 1958  600

> dimnames(wind.compo.sim.block)[-1]

$var
[1] "Fe"    "Al2O3" "SiO2"  "Mn"    "P"     "R"

$sim
  [1] "sim1"   "sim2"   "sim3"   "sim4"   "sim5"   "sim6"
  [7] "sim7"   "sim8"   "sim9"   "sim10"  "sim11"  "sim12"
 [13] "sim13"  "sim14"  "sim15"  "sim16"  "sim17"  "sim18"
 [19] "sim19"  "sim20"  "sim21"  "sim22"  "sim23"  "sim24"
 [25] "sim25"  "sim26"  "sim27"  "sim28"  "sim29"  "sim30"
 [31] "sim31"  "sim32"  "sim33"  "sim34"  "sim35"  "sim36"
 [37] "sim37"  "sim38"  "sim39"  "sim40"  "sim41"  "sim42"
 [43] "sim43"  "sim44"  "sim45"  "sim46"  "sim47"  "sim48"
 [49] "sim49"  "sim50"  "sim51"  "sim52"  "sim53"  "sim54"
 [55] "sim55"  "sim56"  "sim57"  "sim58"  "sim59"  "sim60"
 [61] "sim61"  "sim62"  "sim63"  "sim64"  "sim65"  "sim66"
 [67] "sim67"  "sim68"  "sim69"  "sim70"  "sim71"  "sim72"
```

```
 [73] "sim73"  "sim74"  "sim75"  "sim76"  "sim77"  "sim78"
 [79] "sim79"  "sim80"  "sim81"  "sim82"  "sim83"  "sim84"
 [85] "sim85"  "sim86"  "sim87"  "sim88"  "sim89"  "sim90"
 [91] "sim91"  "sim92"  "sim93"  "sim94"  "sim95"  "sim96"
 [97] "sim97"  "sim98"  "sim99"  "sim100"
```

The result is a new "`DataFrameStack`" with 1958 blocks, 6 variables and 100 realisations, to be plotted with analogue chunks to those used in pages 175 or 212:

```
> myfun = function(i,j){
+   bks = quantile(wind.acomp[,j], prob=(0:10)/10)
+   out = coordinates(gtB) %>%
+   cbind(getStackElement(wind.compo.sim.block,i)) %>%
+     image_cokriged(ivar=j, legendPos = "top", breaks = bks)
+   points(wind.coords, pch=21, col=1, cex=0.75,
+     bg=out$col[cut(wind.acomp[,j],out$breaks)])
+ }
> manipulate(myfun(i,j),
+            i=slider(1, max = 100),
+            j=picker("Fe", "Al2O3", "SiO2", "Mn", "P", "R"))
```

11.3 Case Study: Tellus

To continue the study of the direct sampling simulations obtained in Chap. 10 for the Tellus data in the subcomposition MgO-Al_2O_3-CaO-Fe_2O_3-Rest, we start by uploading the input parameters, training image and realisations we stored in page 197:

```
> # load TI, sampling grid and input parameters:
> load("DS4CoDaTellusInput.RData")
> # load realisations of the DS
> load("DS4CoDaTellusSim.RData")
```

We can now compute local summaries of the realisations, like compositional E-type estimates or total variations, and represent them in maps:

```
> tellusS.dsim.Etype = tellusS.dsim@data %>%
+   gmApply(c("loc","var"), geometricmean) %>% clo
> tellusS.dsim.totvar = tellusS.dsim@data %>%
+   gmApply("loc", function(x) mvar(acomp(x)) )
> out = SpatialGridDataFrame(grid = tellus.gt,
+            data=data.frame(totvar=tellusS.dsim.totvar)) %>%
+     image_cokriged(ivar="totvar", legendPos = "top",
+                    legendPropSpace = 0.12)
> myfun = function(j){
+   bks = quantile(tellus.TI[,j]/100, prob=(0:10)/10, na.rm=T)
+   out =  SpatialGridDataFrame(grid = tellus.gt,
+                           data.frame(tellusS.dsim.Etype)) %>%
+     image_cokriged(ivar=j, legendPos = "top", breaks = bks,
+                    legendPropSpace = 0.12)
+ }
```

```
> sapply(c("MgO", "Al2O3", "CaO", "Fe2O3", "Rest"), myfun)
```

The resulting summary maps are shown in Fig. 11.9. Note that each E-type estimate is given a colour legend established from the training image (Fig. 10.1), to enhance comparability.

Marginal distributions of the simulations can also be compared with the distribution of the input data or of the training image. QQ-plots comparing the distributions of the realisations and the input data are shown in Fig. 11.10. These include as well a QQ-plot of the distribution of each variable on the training image vs. the input data. As in Sect. 11.1.4, an ad hoc function needs to be defined that does the calculations for one variable at one simulation

```
> myQQfun = function(x, j="Rest", factor=100){
+   erg = qqplot(plot.it = F, x= na.omit(tellusS.compo[,j]),
+                y = na.omit(x[,j]*factor)  )
+   return(data.frame(erg))
+ }
```

and then feeds the result to `swarmPlot`

```
> swarmPlot(tellusS.dsim@data, PLOTFUN = myQQfun, j="Rest",
+           .plotargs = list(asp=1, xlab="observed quantiles",
+           ylab="simulation quantiles", main="Rest") )
> abline(a=0, b=1, col=2)
> tellus.TI %>% myQQfun(factor=1) %>% as.matrix %>% lines(col=4)
```

to produce the whole swarm. Figure 11.10 presents the results for all variables, showing very good reproductions for MgO and Al_2O_3, and somewhat less satisfactory reproduction for CaO and Fe_2O_3—which show a branched upper tail—and for Rest—with its branched lower tail. However, the realisations that diverge from the data show spatial distributions consistent with the training image.

Finally, the Tellus data set allows us to show the simulation based accuracy of direct sampling, because in this case the training image happens to provide the underlying true values as well. This is a highly uncommon situation, indeed an undesirable one, as it tends to provide an overoptimistic picture of the goodness of algorithms. Nevertheless, the chunk below can be used to compute both the univariate marginal accuracy for each original variable, and the global Aitchison–Mahalanobis based accuracy for the whole composition

```
> par(mfrow=c(2,3))
> tellus.mask = as.array(tellusS.dsim@data) %>%
+                                 is.na %>% gmApply(1, any)
> tellusS.dsim.masked = setMask(tellusS.dsim@data, !tellus.mask)
> truth = tellus.TI.spdf@data[!tellus.mask,]/100
> for(j in c("MgO", "Al2O3", "CaO", "Fe2O3", "Rest"))
+   tellusS.dsim.masked %>%
+   accuracy(observed = truth, method="simulation", vars=j) %>%
+   plot(main=j)
> tellusS.dsim.masked %>% gmApply(FUN=alr) %>%
+   accuracy(observed = alr(truth)) %>%
+   plot(main="global")
```

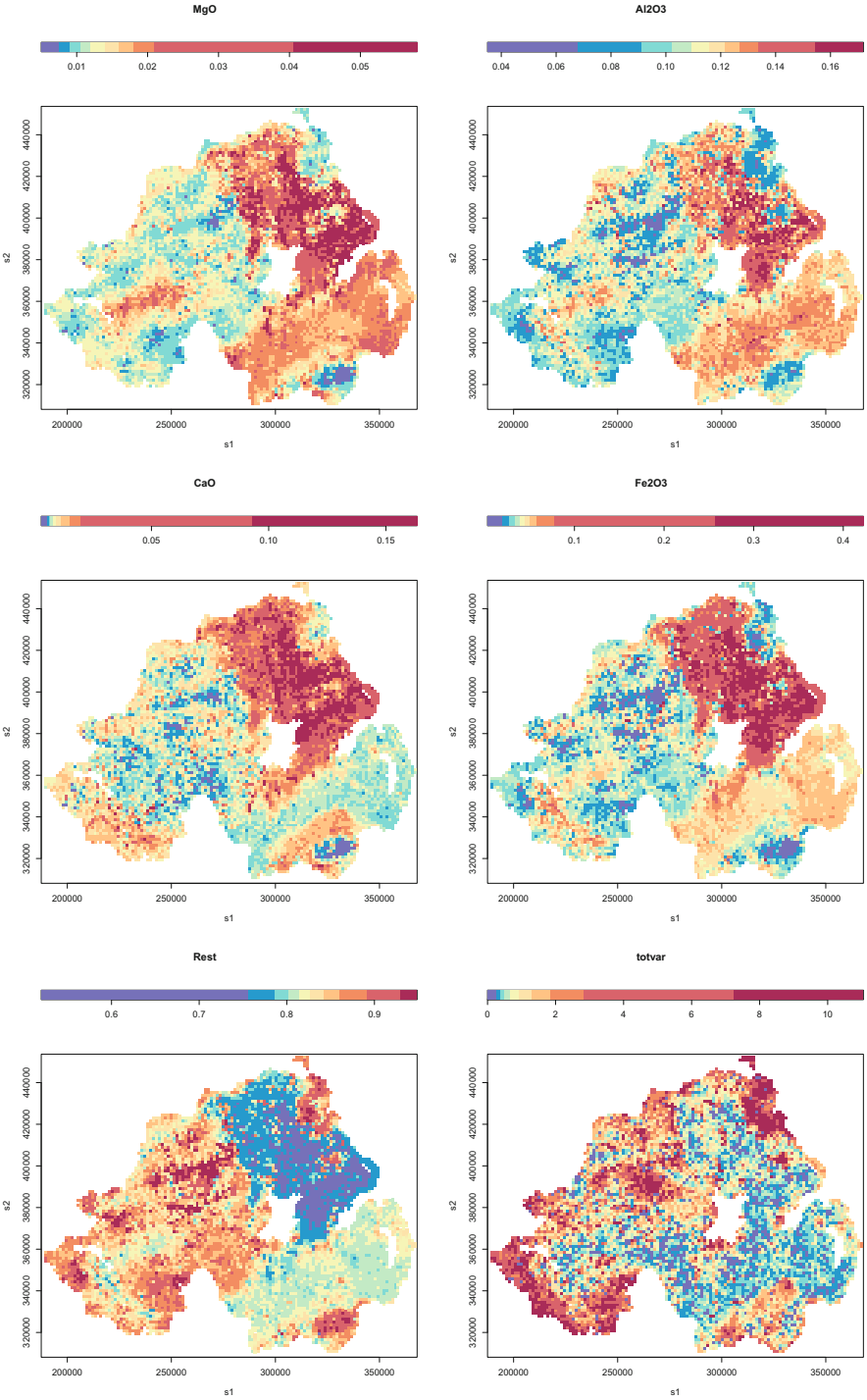

Fig. 11.9 E-type estimates and total variance of the subcomposition for the Tellus data set, out of 25 direct sampling simulations

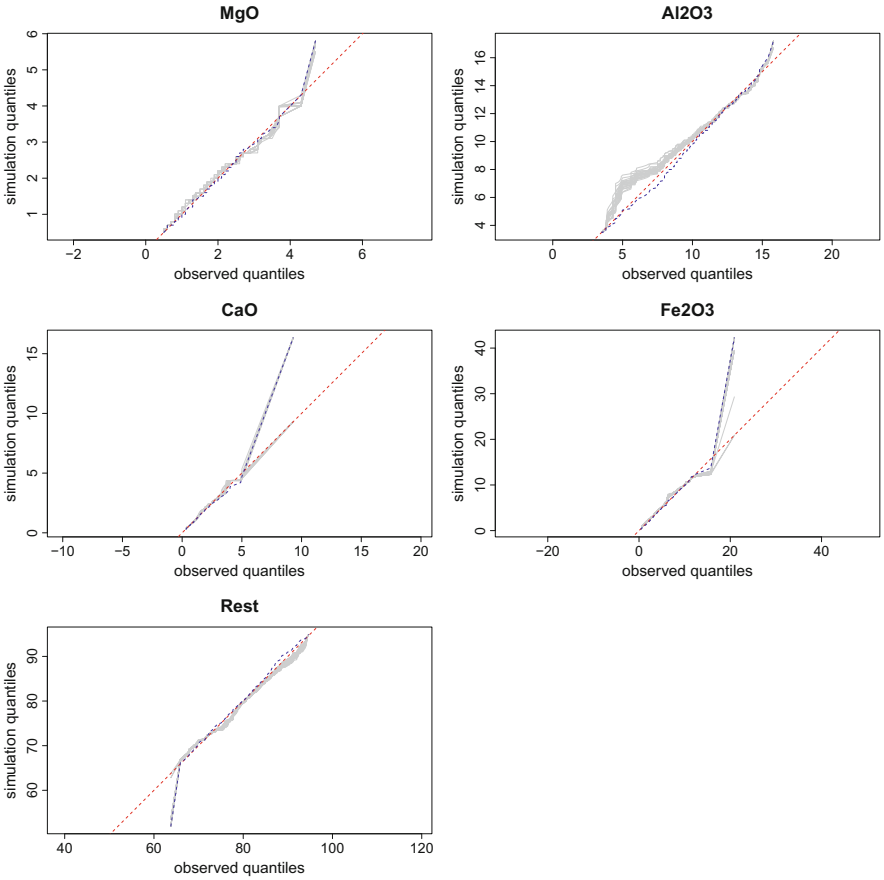

Fig. 11.10 Swarms of QQ-plots of the realisations (grey lines) of the five elements of the Tellus subcomposition and of the training image (blue lines), against the distribution on the conditioning data (red reference line)

Here we need to start by creating a mask that excludes both grid nodes outside the training image and locations that are taken as conditioning data. With it, we extract, from all target locations, both the simulated realisations and the true values (these last normalised to sum to 1). Then we compute and plot the accuracy for each variable, and finally for all together through an alr transformation. The result is shown in Fig. 11.11, resulting in a very optimistic picture: MgO is slightly inaccurate (representing slightly lower variability than desired, as the curve plots below the reference line), while the rest of the variables show a virtually ideal behaviour. Finally, the compositional accuracy offers also a reasonable picture, with a notable overdispersion of simulations in the log-ratio geometry.

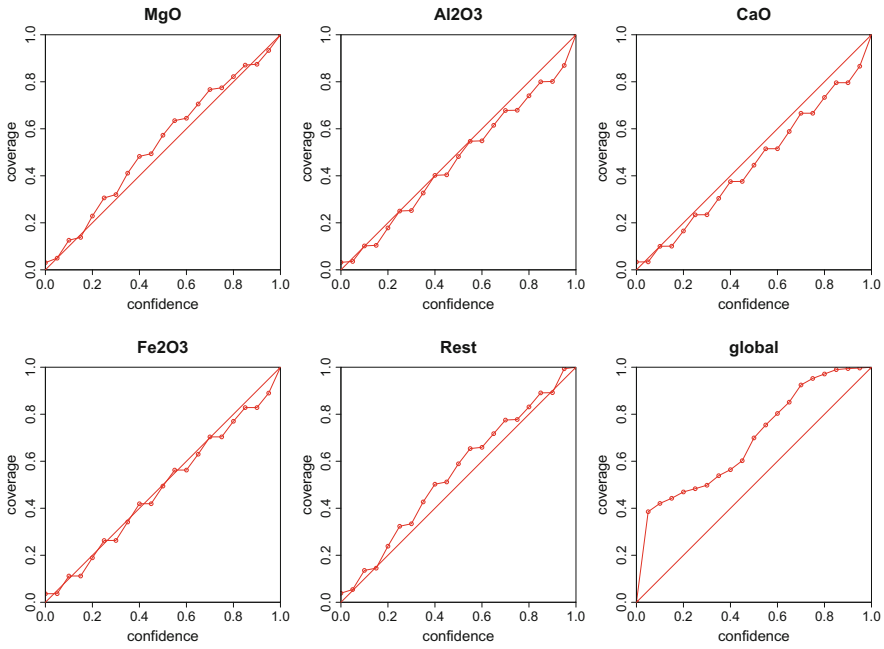

Fig. 11.11 Direct sampling accuracy plots for each original variable and for the whole composition of the Tellus subcomposition

Problems

11.1 Evaluation of Direct Sampling Simulation of Windarling East

Take the direct sampling simulations for the Eastern bench of Windarling that we stored in file "`Windarling_TIsim_East.RData`" at the end of Sect. 10.5. Compute and represent:

(a) E-type maps of the expected value of the composition and of its total variation
(b) Swarms of QQ-plots of the marginal distribution of each of the six components on the simulations vs. the distribution on the data and vs. the distribution on the training image
(c) Block averages at a block grid of $9 \times 9\text{m}^2$, i.e. with each block formed by 3×3 finer grid nodes
(d) Grade and tonnage curves at the block support; compare with those of Fig. 11.7

11.2 Evaluation of Simulation Results for NGSA Major Elements

For the realisations simulations for the NGSA subcomposition of major elements in the East of Australia, compute and represent:

(a) E-type maps of the expected value of the composition and of its total variation

(b) Swarms of QQ-plots of the marginal distribution of each of the six components on the simulations versus the distribution on the data and versus the distribution on the training image
(c) Block averages at a block grid of $25 \times 25\text{mm}^2$, i.e. with each block formed by 5×5 finer grid nodes
(d) Grade and tonnage curves at the block support; compare with grade-tonnage curves calculated for the cokriging estimates calculated in Problem 6.3

11.3 A Comparison of Results from DS Simulation and SG-Cosimulation of the Tellus Subcomposition

For the realisations of the subcomposition based on the subcomposition MgO-Al_2O_3-CaO-Fe_2O_3-Rest within the Tellus subset generated via SG-cosimulation in Problem 9.1, calculate and represent:

(a) E-type maps of the expected value of the composition and of its total variation
(b) Swarms of QQ-plots of the marginal distribution of each of the six components on the simulations versus the distribution on the data and versus the distribution of the entire Tellus data
(c) Swarms of experimental variograms of the realisations with raw variograms of the data superimposed
(d) Accuracy plots following the same approach as that for constructing Fig. 11.11
(e) Also generate swarms of experimental variograms of the DS realisations with raw variograms of the data superimposed
(f) Compare the results from SG-cosimulation with those from direct sampling simulation

References

Abzalov, M. (2016). *Applied mining geology* (448 pp.). Springer International Publishing.
David, M. (1977). *Geostatistical ore reserve estimation*. Series on developments in geomathematics (Vol. 2, 364 pp.). New York, NY, USA: Elsevier.
Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Applied Geostatistics Series (483 pp.). New York, NY, USA: Oxford University Press.
Lantuéjoul, C. (1990). *Cours de Selectivité* (73 pp.). Centre de Géostatistique, Ecole de Mines de Paris. Fontainebleau: Publication C-140.
Leuangthong, O., McLennan, J. A., & Deutsch, C. V. (2004). Minimum acceptance criteria for geostatistical realisations. *Natural Resources Research, 13*(3), 131–141.
Rossi, M. E., & Deutsch, C. V. (2014). *Mineral resource estimation* (332 pp.). Dordrecht: Springer.

Appendix A
Matrix Decompositions

Matrix decompositions play an important role in multivariate statistics and in geostatistics. The matrices being decomposed are symmetric and positive (semi-)definite, and depending on the application different decompositions are applied. In the following sections, we review the concepts relating to these decompositions. General references are Horn and Johnson (1985), Golub and Loan (2013) and Strang (2013). It is assumed that the reader is familiar with matrix arithmetic.

A.1 LU Decomposition

Given a square matrix $\mathbf{A}$ of size $n \times n$, a decomposition into a product of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$ such that $\mathbf{A} = \mathbf{LU}$ is called an *LU decomposition* of $\mathbf{A}$. This decomposition is not unique. For example,

$$\mathbf{A} = \begin{bmatrix} 3 & 4 \\ 18 & 22 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 6 & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 0 & -2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 18 & -2 \end{bmatrix} \begin{bmatrix} 1 & \frac{4}{3} \\ 0 & 1 \end{bmatrix}$$

and one can construct many more. When the matrix $\mathbf{A}$ is symmetric and positive definite, then there exists a unique lower triangular matrix $\mathbf{R}$ with positive diagonal entries such that $\mathbf{A} = \mathbf{R}\mathbf{R}^t$. This decomposition is known as the *Cholesky decomposition*, and the lower triangular matrix $\mathbf{R}$ is called the *Cholesky factor*. For example, the Cholesky decomposition of $\mathbf{A} = \begin{bmatrix} 9 & -3 \\ -3 & 5 \end{bmatrix}$ is

$$\mathbf{A} = \begin{bmatrix} 9 & -3 \\ -3 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 0 & 2 \end{bmatrix}$$

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3

A.2 Spectral Decomposition

Given a real $n \times n$ matrix $\mathbf{A}$ a scalar λ is said to be an *eigenvalue* of $\mathbf{A}$, if there exists a non-zero vector $\mathbf{v}$ such that $\mathbf{Av} = \lambda\mathbf{v}$. The vector $\mathbf{v}$ is called an *eigenvector* for $\mathbf{A}$. It should be noted that for a general real $n \times n$ matrix, an eigenvalue can be complex, in which case its conjugate is also an eigenvalue and the corresponding eigenvectors are complex conjugates also.

The existence of eigenvalues and eigenvectors of a matrix is relevant for matrix diagonalisation. A matrix $\mathbf{A}$ is said to be diagonalisable, if there exists a matrix $\mathbf{D}$ and an invertible matrix $\mathbf{S}$ such that $\mathbf{A} = \mathbf{SDS}^{-1}$. It can be shown that $\mathbf{A}$ is diagonalisable over $\mathbb{C}$ if and only if $\mathbf{A}$ has n linearly independent eigenvectors.

The matrices of particular interest to this book are real symmetric matrices, and for these the spectral theorem applies.

Theorem A.1 *Given a real symmetric $n \times n$ matrix $\mathbf{A}$, then*

1. *The eigenvalues of $\mathbf{A}$ are real.*
2. *There exists an orthogonal matrix $\mathbf{U}$ such that $\mathbf{U}^t\mathbf{AU}$ is diagonal with the diagonal entries equal to the eigenvalues of $\mathbf{A}$ listed in accordance to their multiplicity.*

For a proof of this theorem, see for example Horn and Johnson (1985). The column vectors of the matrix $\mathbf{U}$ are the eigenvectors of the matrix $\mathbf{A}$ normalised to have unit norm. Given a real symmetric matrix $\mathbf{A}$, it can be written as a sum of rank one matrices as follows:

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^t,$$

where $\mathbf{u}_i$ denotes the ith column of the matrix $\mathbf{U}$ and λ_i the ith eigenvalue of $\mathbf{A}$. For each i, $i = 1, \ldots, n$ the matrix $\mathbf{u}_i\mathbf{u}_i^t$ has rank 1.

Further, when $\mathbf{A}$ is positive semi-definite, the eigenvalues of $\mathbf{A}$ are non-negative and the square root of $\mathbf{A}$ is given by $\mathbf{A}^{1/2} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}\mathbf{U}^t$ where $\boldsymbol{\Lambda}$ denotes the diagonal matrix of $\mathbf{A}$ corresponding to $\mathbf{U}$.

A.3 Generalised Eigenvalue Problem

In the previous section, we considered the standard eigenvalue problem $\mathbf{Av} = \lambda\mathbf{v}$. The *generalised eigenvalue problem* is closely related and may be stated as follows: Given a matrix $\mathbf{A}$ and an invertible matrix $\mathbf{B}$, find λ and $\mathbf{v}$, $\mathbf{v} \neq \mathbf{0}$ such that the equation $\mathbf{Av} = \lambda\mathbf{Bv}$ is satisfied.
The pair $(\lambda, \mathbf{v})$ then constitutes a generalised eigenpair for $(\mathbf{A}, \mathbf{B})$. The generalised eigenvalue problem may be rewritten as a standard eigenvalue problem by premul-

tiplying both sides of the equation $\mathbf{Av} = \lambda\mathbf{Bv}$ by $\mathbf{B}^{-1}$ to obtain $\mathbf{B}^{-1}\mathbf{Av} = \lambda\mathbf{v}$. If the matrices $\mathbf{B}$ and $\mathbf{A}$ are positive definite and positive semi-definite respectively, then an often more convenient method to rewrite a generalised eigenvalue problem as a standard one is to first determine the Cholesky factor $\mathbf{R_B}$ of $\mathbf{B}$. Then putting $\mathbf{A_B} = {\mathbf{R_B}}^{-1}\mathbf{A}{\mathbf{R_B}}^{-t}$ and $\mathbf{v_B} = {\mathbf{R_B}}^{t}\mathbf{v}$, we obtain $\mathbf{A_B v_B} = \lambda\mathbf{v_B}$. Alternatively we can make use of the square root of the matrix $\mathbf{B}$ as determined above: $\mathbf{A_S v_S} = \lambda\mathbf{v_S}$ where $\mathbf{A_S} = \mathbf{S}^{-1}\mathbf{A}\mathbf{S}^{-1} = \mathbf{S}^{-1}\mathbf{A}\mathbf{S}^{-t}$ and $\mathbf{v_S} = \mathbf{Sv}$ where $\mathbf{S} = \mathbf{U_B}\boldsymbol{\Lambda}^{1/2}{\mathbf{U_B}}^{t}$. It should be noted that if $\mathbf{A}$ is symmetric, then both $\mathbf{A_S}$ and $\mathbf{A_B}$ are symmetric and so methods for determining the eigenvalues and eigenvectors of a symmetric matrix can be applied.

The generalised eigenvalue problem is relevant for the calculation of Minimum/Maximum Autocorrelation factors (MAF; Sect. 4.4).

A.4 Singular Value Decomposition

The *singular value decomposition* (SVD) is a matrix decomposition which is of particular use when wishing to find a factorisation for a non-square matrix $\mathbf{A}$. We will assume that $\mathbf{A}$ has size $m \times n$.

The SVD of $\mathbf{A}$ is the product $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^t$ where $\boldsymbol{\Sigma}$ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i, i = 1, \ldots, p \leq \min(n, m)$ and 0s elsewhere. The matrices $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$ are orthogonal matrices of size $m \times m$ and $n \times n$, respectively. The values $\sigma_1, \sigma_2, \ldots, \sigma_p$ are called the *singular values*. They are by convention non-negative and arranged in descending order. The column vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are called the *left singular vectors* and *right singular vectors*, respectively. In the case where $m \geq n$ they satisfy

$$\mathbf{A}^t\mathbf{u}_i = \sigma_i\mathbf{v}_i\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i$$

for $i = 1, \ldots, n$, as a direct consequence we then have

$$\mathbf{A}\mathbf{A}^t\mathbf{u}_i = \sigma_i^2\mathbf{u}_i \tag{A.1}$$

$$\mathbf{A}^t\mathbf{A}\mathbf{v}_i = \sigma_i^2\mathbf{v}_i \tag{A.2}$$

for $i = 1, \ldots, n$.

One of the applications of SVD is the computation of principal components and the construction of biplots (Sect. 3.3). Another common application is the calculation of the *pseudoinverse* of $\mathbf{A}$. Given the factorisation $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^t$, we define $\boldsymbol{\Sigma}^-$ to be the $n \times m$ matrix with entries $\boldsymbol{\Sigma}^-_{ii} = \frac{1}{\sigma_i}$ for $1 \leq i \leq p$ and $\boldsymbol{\Sigma}^-_{ij} = 0$ else. Then the pseudoinverse $\mathbf{A}^-$ of $\mathbf{A}$ is defined as

$$\mathbf{A}^- = \mathbf{V}\boldsymbol{\Sigma}^+\mathbf{U}^t.$$

The matrix $\mathbf{A}^-$ has size $n \times m$ and

$$\mathbf{A}^-\mathbf{A} = \mathbf{V}\boldsymbol{\Sigma}^-\mathbf{U}^t\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^t = \mathbf{V}\text{diag}(\left[\mathbf{1}_p^t, \mathbf{0}_{n-p}^t\right])\mathbf{V}^t \tag{A.3}$$

$$\mathbf{A}\mathbf{A}^- = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^t\mathbf{V}\boldsymbol{\Sigma}^+\mathbf{U}^t = \mathbf{U}\text{diag}(\left[\mathbf{1}_p^t, \mathbf{0}_{m-p}^t\right])\mathbf{U}^t. \tag{A.4}$$

So in general both $\mathbf{A}^-\mathbf{A}$ and $\mathbf{A}\mathbf{A}^-$ are projection matrices, and they project onto the row space and column space of $\mathbf{A}$, respectively. Assuming that $p = \min(m, n) = n$, then $\mathbf{A}\mathbf{A}^- = \mathbf{U}\text{diag}(\left[\mathbf{1}_p^t, \mathbf{0}_{m-p}^t\right])\mathbf{U}^t$ and $\mathbf{A}^-\mathbf{A} = \mathbf{I}_n$.

References

Golub, G. H., & Loan, C. F. V. (2013). *Matrix computations, fourth edition* (756 pp.). Baltimore, MD, USA: The Johns Hopkins University Press.

Horn, R. A., & Johnson, C. R. (1985). *Matrix analysis* (561 pp.). New York, NY, USA: Cambridge University Press.

Strang, G. (2013). *Linear algebra and its applications, fourth edition* (488 pp.). Belmont, CA, USA: Thomson Brooks/Cole.

Appendix B
Complete Data Analysis Workflows

In this appendix, workflows for the various aspects of a geostatistical analysis will be presented for quick reference. There are in essence five components, some of which are always performed when conducting a geostatistical analysis of compositional data. These are

1. Exploratory data analysis
2. Modelling the spatial continuity and validating the model
3. Estimation
4. Simulation
5. Postprocessing

In the following section, the main components will be listed together with the relevant functions.

B.1 Exploratory Data Analysis

Exploratory data analysis comprises several parts that are critical for gaining a thorough understanding of the characteristics of the data one is dealing with. It can be split into numerical and spatial EDA.

B.1.1 Numerical EDA

The assumption we will make is that the variables have been selected and declared to be compositions.

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3

1. Visual inspection of the data via
   - Harker diagrams (`plot`, `pairs`)
   - Ternary or quaternary plots (`plot.acomp`, `plot3D.acomp`)
   - Pairwise density plots (`pairs` with panel `vp.lrdensityplot`)
   - Histograms and boxplots of pairwise ratios (`boxplot.acomp`)
2. Calculation of compositional statistics (`mean`, `variation`, `summary`)
3. Calculation of PCA and biplots (`princomp`, `screeplot`, `coloredBiplot`)
4. Normality checks (`qqnorm.acomp`, `MVN`)
5. Outlier checks (`OutlierClassifier1`)

The code for these calculations makes use of "`compositions`" and "`gmGeostats`". Detailed explanations can be found in Chap. 3.

```
> #
> # data selection
> #
> data("Windarling",package="gmGeostats")
> windarling = Windarling
> wind.compo = dplyr::select(windarling, Fe:LOI)
> wind.compo$R = 1-rowSums(wind.compo)
> #
> # quick diagrams
> #
> wind.compo %>% clr %>% data.frame %>%
+   dplyr::select(Fe, SiO2, Al2O3, R) %>%
+      plot(panel=vp.kde2dplot)
> wind.compo %>% dplyr::select(Fe, SiO2, Al2O3, R) %>%
+   acomp %>% pairs
> wind.compo %>% dplyr::select(Fe, SiO2, Al2O3, R) %>%
+   acomp %>% boxplot
> #
> # numerical descriptive statistics
> #
> (g = mean(acomp(wind.compo)))
> clr(g)
> variation(acomp(wind.compo))
> #
> # PCA and biplots
> #
> pc = wind.compo %>% clr %>% princomp
> names(pc)
> (propvar=cumsum(pc$sdev^2)/sum(pc$sdev^2))
> screeplot(pc, space=0)
> lines(propvar, lwd=2, col=2, type="o", pch=19)
> abline(h=c(0.95,1), lty=2, col=2)
> coloredBiplot(pc, choices = 1:2, xlabs.pc=1, cex=c(0.5,1),
+               col=1, xlabs.col=windarling$Lithotype)
> legend("bottomright", fill=1:4,
+        legend=levels(windarling$Lithotype))
> #
> # selected ternary diagrams according to biplot
```

```
> #
> par(mfrow=c(1,2))
> wind.compo %>% dplyr::select(Mn, Al2O3, SiO2) %>% acomp %>%
+   plot(center=T, pca=T, col.pca=1, col=windarling$Lithotype)
> wind.compo %>% dplyr::select(P, R, Fe) %>% acomp %>%
+   plot(center=T, pca=T, col.pca=1, col=windarling$Lithotype)
> #
> # normality checks
> #
> wind.compo %>% dplyr::select(Fe, SiO2, Al2O3, R) %>%
+   acomp %>% qqnorm
> #
> # robust statistics
> #
> round(mean(acomp(wind.compo), robust="mcd"), dig=3)
> round(variation(acomp(wind.compo), robust="mcd"), dig=3)
> #
> # outliers (in subcompositions, to save time)
> #
> wind.subcompo = wind.compo %>%
+   dplyr::select(Fe, SiO2, Al2O3, Mn) %>% acomp
> wind.cf1 = wind.subcompo %>% ClusterFinder1
> sort(wind.cf1$typeTbl, decreasing = T)
> wind.subcompo = wind.compo %>%
+   dplyr::select(Fe, SiO2, Al2O3, Mn) %>% acomp
> wind.cf1 = wind.subcompo %>% ClusterFinder1
> sort(wind.cf1$typeTbl, decreasing = T)
> wind.oc1g = OutlierClassifier1(wind.subcompo, type="grade")
> table(wind.oc1g, windarling$Lithotype)
> wind.oc1b = OutlierClassifier1(wind.subcompo, type="best")
> table(wind.oc1b, windarling$Lithotype)
> wind.oc1a = OutlierClassifier1(wind.subcompo, type="all")
> table(wind.oc1a, windarling$Lithotype)
```

B.1.2 Spatial EDA

Just as the numerical EDA there are several steps related to a spatial EDA of the data. These are

1. Spatial maps of the data locations coloured by the variable of interest (`pairsmap`).
2. Swath plots to investigate the behaviour of the variable of interest in the spatial coordinate directions and to detect trends in the data (`swath`, `swath.acomp`).
3. Experimental direct and cross variograms to start understanding the spatial continuity of the variables of interest in several compositional coordinate representations, at a minimum, in pwlr and one of alr, clr or ilr form. (pwlr: `logratioVariogram`; other logratios: `gstat` or `make.gmCompositional-GaussianSpatialModel`, followed by `variogram`).

4. Variogram maps to check for the presence of anisotropy (`logratio Variogram`).
5. The calculation of MAF factors and construction of MAF biplots and associated checks of spatial decorrelation through the transformation to factors (`Maf`).

The code for these calculations makes use of "`compositions`", "`gstat`" and "`gmGeostats`". Detailed explanations can be found in Chap. 4.

```
> #
> # Definition of subcomposition used from hereon and
> # spatial coordinates
> #
> data("Windarling",package="gmGeostats")
> windarling0=Windarling
> windarling = windarling0 %>%
>           dplyr::filter(Lithotype=="hematite"|Lithotype=="goethite")
> wind.compo = windarling %>%
>           dplyr::select(c("Fe","Al2O3","SiO2","P","Mn"))
> wind.compo$R = 1-rowSums(wind.compo)
> wind.coords = as.matrix(windarling[,c("Easting","Northing")])
> #
> # Construction of spatial maps (raw and log-ratios)
> #
> pairsmap(loc=wind.coords, data=wind.compo, mfrow=c(6,1),
+          cexrange=c(1,3)*0.5, foregroundcolor=NA)
> pairsmap(loc=wind.coords, data=alr(wind.compo), mfrow=c(5,1),
+          cexrange=c(1,3)*0.5, foregroundcolor=NA)
> #
> # Construction of swath plots in pwlr coordinates
> #
> pwlrSwath(wind.coords[,"Easting"], wind.compo, col=8,
+           xlab="Easting")
> pwlrSwath(wind.coords[,"Northing"], wind.compo, col=8,
+           xlab="Northing")
> #
> # Calculation of omnidirectional experimental variograms
> #
> wind.pwlr.vg = logratioVariogram(comp=acomp(wind.compo),
+                        loc=wind.coords, maxdist=50, nbins=15)
> plot(wind.pwlr.vg)
> #
> # recast to alr
> #
> wind.alr.vg = as.gstatVariogram(wind.pwlr.vg, V = "alr")
> plot(wind.alr.vg)
> #
> # Calculation of variation-variogram maps
> #
> wind.pwlr.vgAnis = logratioVariogramAnisotropy(
+   comp=acomp(wind.compo),
+   loc=wind.coords, maxdist=50, nbins=10,
+   azimuth.tol=22.5, azimuths=10*(0:35))
> image(wind.pwlr.vgAnis)
> #
```

```
> # Calculation and plot of directional experimental variograms
> #
> plot(wind.pwlr.vgAnis, azimuths= c(0,90),lty=1)
> #
> # Calculation of MAF factors and MAF biplot
> #
> maf.wind = Maf(acomp(wind.compo), vg=wind.pwlr.vg, i=2)
> coloredBiplot(maf.wind, xlabs.pc=19, xlabs.col=
+               as.integer(windarling$Lithotype),
+               cex=c(0.5,1), col="black")
> #
> # Check of spatial decorrelation
> #
> wind.maf.gg = make.gmCompositionalGaussianSpatialModel(
+   data = wind.compo, coords=wind.coords, V=maf.wind$loadings,
+   prefix="maf")
> wind.maf.vg = variogram(wind.maf.gg, cutoff=50, width=3.5)
> plot(wind.maf.vg)
> g = spatialDecorrelation(wind.maf.vg, method=c("add","rdd"))
> mean(g) # average values
> plot(g) # profile
> g = spatialDecorrelation(wind.maf.vg, wind.alr.vg, method="sde")
> mean(g)
> plot(g)
```

B.2 Modelling the Spatial Continuity

Once the EDA has been completed, the next step in a geostatistical analysis is to build models of the spatial continuity. Since the data considered here are compositional, unless a factorial representation has been derived based on one of the methods in Sect. 5.4, this requires the fitting of an LMC. When a suitable model cannot be found directly, it is possible to use MAF to construct the model step-wise.

Once the LMC has been fitted, its adequacy for estimation needs to be established, which is done via multivariate cross-validation based on a "unique neighbourhood". This process can also be used to establish which model to accept, if there are several candidate models.

In brief, we have the following steps:

1. Determine if the data can be deemed to be stationary, this requires checking the experimental variograms to detect and underlying systematic trends (`logratioVariogram` or `variogram`, then `plot`).
2. Build an LMC (with real data, use `vgm`; with compositions use `CompLinMod` `CoReg` or `LMCAnisCompo` if anisotropy is present) and fit it to the composition in log-ratio coordinates, ensuring to account for anisotropy (`fit_lmc`).
3. If the data exhibit a deterministic trend, fit an isotropic LMC to the composition.

4. Validate the LMC via multivariate cross-validation based on all sample locations and the appropriate estimator, i.e. Ordinary Cokriging or Universal Cokriging ("gstat": `gstat.cv`).
5. Among the candidate models, choose the model with the best cross-validation performance to use in estimation or simulation.

The code for these calculations makes use of "`compositions`", "`gstat`" and "`gmGeostats`". Detailed explanations can be found in Chap. 5.

```
> #
> # Fitting of an LMC to the omnidirection experimental direct
> # and cross variograms in alr coordinates
> #
> wind.alr.gg = make.gmCompositionalGaussianSpatialModel(
+   data=wind.compo, coords=wind.coords, V="alr")
> wind.alr.vg = variogram(wind.alr.gg, cutoff=50, width=3.5)
> # A first trial
> md0=vgm("Exp")
> wind.alr.gg = fit_lmc(v=wind.alr.vg, g=wind.alr.gg, model=md0)
> # Improved fit
> plot(wind.alr.vg, wind.alr.gg$model)
> md1 = vgm(psill=1, model="Sph", range=15, nugget=1)
> md1 = vgm(psill=1, model="Sph", range=45, add.to=md1 )
> wind.alr.gg = fit_lmc(v=wind.alr.vg, g=wind.alr.gg, model=md1,
+                       correct.diagonal = 1.001)
> plot(wind.alr.vg, wind.alr.gg$model)
> #
> # Calculation and fit of LMC for directional experimental
> # direct and cross variograms
> #
> wind.alr.vgAnis = variogram(wind.alr.gg, cutoff=50, width=5,
+                             alpha=c(90,180), tol.hor=45)
> plot(wind.alr.vgAnis,group.id=FALSE,col=c("red","black"))
> #
> # If the fit above is not satisfactory, derivation of a
> # suitable LMC from MAF factors
> #
> wind.maf.ggAnis = make.gmCompositionalGaussianSpatialModel(
+                       data = wind.compo, coords = wind.coords,
+                       V = maf.wind$loadings)
> #
> # definition of gstat containers for the MAF factors and
> # calculation of directional experimental variograms
> #
> wind.maf.sdf = data.frame(cbind(wind.coords, maf.wind$scores))
> wind.maf1.gg = gstat(id="maf1", locations=~Easting+Northing,
+         formula=maf1~1, data=wind.maf.sdf)
> wind.maf1.vg = variogram(wind.maf1.gg, cutoff=50,
+                      width=3.5, alpha=c(90,180))
> wind.maf2.gg = gstat(id="maf2", locations=~Easting+Northing,
+         formula=maf2~1, data=wind.maf.sdf)
> wind.maf2.vg = variogram(wind.maf2.gg, cutoff=50,
+                      width=3.5, alpha=c(90,180))
```

```
> wind.maf3.gg = gstat(id="maf3", locations=~Easting+Northing,
+         formula=maf3~1, data=wind.maf.sdf)
> wind.maf3.vg = variogram(wind.maf3.gg, cutoff=50,
+                      width=3.5, alpha=c(90,180))
> wind.maf4.gg = gstat(id="maf4", locations=~Easting+Northing,
+         formula=maf4~1, data=wind.maf.sdf)
> wind.maf4.vg = variogram(wind.maf4.gg, cutoff=50,
+                      width=3.5, alpha=c(90,180))
> wind.maf5.gg = gstat(id="maf5", locations=~Easting+Northing,
+         formula=maf5~1, data=wind.maf.sdf)
> wind.maf5.vg = variogram(wind.maf5.gg, cutoff=50,
+                      width=3.5, alpha=c(90,180))
> #
> # Fitting veriogram models to experimental variograms
> #
> wind.maf1.md = vgm(nugget=0.15, psill =0.2, range=10,
+           model = "Sph", anis=c(90, 1))
> wind.maf1.md = vgm(add.to=wind.maf1.md, psill =0.7, range=60,
+            model = "Sph", anis=c(90, 0.6))
> wind.maf2.md = vgm(nugget=0.2, psill =0.25, range=10,
+           model = "Sph", anis=c(90, 1))
> wind.maf2.md = vgm(add.to=wind.maf2.md, psill =0.5, range=70,
+           model = "Sph", anis=c(90, .8))
> wind.maf3.md = vgm(nugget=0.3, psill =0.25, range=10,
+           model = "Sph", anis=c(90, 1))
> wind.maf3.md = vgm(add.to=wind.maf3.md, psill =0.45, range=40,
+           model = "Sph", anis=c(90, 0.5))
> wind.maf4.md = vgm(nugget=0.3, psill =0.25, range=10,
+           model = "Sph", anis=c(90, 1))
> wind.maf4.md = vgm(add.to=wind.maf4.md, psill =0.35, range=25,
+           model = "Sph", anis=c(90, 0.5))
> wind.maf4.md = vgm(add.to=wind.maf4.md, psill =0.10, range=40,
+           model = "Sph", anis=c(90, .5))
> wind.maf5.md = vgm(nugget=0.35, psill =0.25, range=10,
+           model = "Sph", anis=c(90, 1))
> wind.maf5.md = vgm(add.to=wind.maf5.md, psill =0.25, range=25,
+            model = "Sph", anis=c(90, .5))
> wind.maf5.md = vgm(add.to=wind.maf5.md, psill =0.20, range=40,
+           model = "Sph", anis=c(90, .5))
> #
> # Definition of gstat containers with models
> #
> wind.maf1.gg = gstat(id="maf1", locations=~Easting+Northing,
+        formula=maf1~1, data=wind.maf.sdf, model =wind.maf1.md)
> wind.maf2.gg = gstat(id="maf2", locations=~Easting+Northing,
+        formula=maf2~1, data=wind.maf.sdf, model =wind.maf2.md)
> wind.maf3.gg = gstat(id="maf3", locations=~Easting+Northing,
+        formula=maf3~1, data=wind.maf.sdf, model =wind.maf3.md)
> wind.maf4.gg = gstat(id="maf4", locations=~Easting+Northing,
+        formula=maf4~1, data=wind.maf.sdf, model =wind.maf4.md)
> wind.maf5.gg = gstat(id="maf5", locations=~Easting+Northing,
+        formula=maf5~1, data=wind.maf.sdf, model =wind.maf5.md)
> #
> # Extraction of MAF variogram model parameters from containers
```

```
> #
> # make a matrix with all (diagonal) sills:
> MAFm=matrix(0,ncol=5,nrow=6)
> rownames(MAFm)=c("Sp0","Sp10","Sp25","Sp40","Sp60","Sp70")
> aux=list(wind.maf1.gg, wind.maf2.gg, wind.maf3.gg,
+          wind.maf4.gg, wind.maf5.gg)
> for (i in 1:5) {
+   MAFm[paste("Sp",aux[[i]]$model[[1]]$range,sep=""),i]=
+     aux[[i]]$model[[1]]$psill}
> # make a matrix with all anisotropic ranges
> MAFanis=matrix(0,ncol=5,nrow=6)
> rownames(MAFanis)=c("Sp0","Sp10","Sp25","Sp40","Sp60", "Sp70")
> for (i in 1:5) {
+   MAFanis[paste("Sp",aux[[i]]$model[[1]]$range,sep=""),i]=
+     aux[[i]]$model[[1]]$anis1
+   }
> # re-dimensionalize sills and ranges
> MAFsills=apply(MAFm,1,diag)
> dim(MAFsills)=c(5,5,6)
> MAFranges=cbind(c(0,10,25,40, 60, 70),c(0,10,12.5,20,36,49))
> MAFazimuth=c(0,0,90,90,90,90)
> # construction of model from pairwise log-ratios from MAF
> # variogram model parameters
> wind.pwlr.mdAnis=LMCAnisCompo(wind.compo,
+           models=c("nugget", "sph", "sph","sph","sph","sph"),
+           azimuths=MAFazimuth, ranges=MAFranges,
+           sillarray=MAFsills, V=maf.wind$loadings, tol=1e-12)
> # Plot of resultant LMC for alr data
> wind.alr.mdAnis=as.variogramModel(wind.pwlr.mdAnis,V="alr")
> variogramModelPlot(wind.alr.vgAnis, wind.alr.mdAnis)
```

The following code deals with validation, either for variogram models or for neighbourhood selection. Detailed information can be found in Chap. 7. The first part covers leave one out cross-validation, the second part training–validation subset splitting.

```
> #
> # 1.- Cross-validation, without separate validation set
> #
> wind.alr.xvAnis= gstat.cv(wind.alr.ggAnis,remove.all=TRUE,
+                           all.residuals=TRUE,verbose=FALSE)
> xv.true = data.frame(alr(wind.compo))
> xv.res = wind.alr.xvAnis
> xv.preds = xv.true-xv.res
> summary(xv.res)
> #
> # Marginal diagnostic plots
> #
> par(mfcol=c(4,5), mar=c(4,4,3,1))
> for(i in 1:5){
+   # predicted vs observed
+   plot(xv.true[,i]~xv.preds[,i], ylab="observed",
+        xlab="predicted", asp=1, main=colnames(xv.preds)[i])
+   abline(a=0, b=1)
```

```
+   abline(lm(xv.true[,i]~xv.preds[,i]), col=2)
+   # residual histogram
+   hist(xv.res[,i], xlab="residual", main="")
+   # residual QQ plot
+   qqnorm(xv.res[,i], main="normal QQ plot")
+   qqline(xv.res[,i], col=2)
+   abline(h=0)
+   # residual vs covariable plot
+   boxplot(xv.res[,i]~windarling$Lithotype,
+           horizontal=T)
+   abline(v=0)
+ }
> #
> # PWLR diagnostic plots
> #
> xv.preds.comp = alrInv(xv.preds, orig=wind.compo)
> xv.true.comp = alrInv(xv.true, orig=wind.compo)
> xv.res.comp = alrInv(xv.res, orig=wind.compo)
> par(mfrow=c(6,6), mar=c(1,1,1,1)/4, oma=c(3,3,3,3))
> for(i in 1:6){
+   for(j in 1:6){
+     if(i==j){
+       plot(c(0,1), c(0,1), bty="n",xaxs="i", yaxs="i",
+            xaxt="n", yaxt="n", pch="")
+       text(0.5, 0.5, labels =
+              expression(Fe, Al[2]*O[3],
+              SiO[2], Mn, P , Rest)[i], cex=1.25)
+     }else if(i>j){
+       # lower tri: residual uncorrelated with prediction?
+       x = log(xv.preds.comp[,i]/xv.preds.comp[,j])
+       y = log(xv.res.comp[,i]/xv.res.comp[,j])
+       vp.kde2dplot(x,y, add=F, colpalette = spectralcolors)
+       abline(h=0)
+     }else{
+       # upper tri: prediction correlated with observed?
+       x = log(xv.preds.comp[,i]/xv.preds.comp[,j])
+       y = log(xv.true.comp[,i]/xv.true.comp[,j])
+       vp.kde2dplot(x,y, add=F, colpalette = spectralcolors)
+       abline(a=0, b=1)
+     }
+   }
+ }
> mtext(side = 1:4, line=1, outer=TRUE, text=paste("log-ratio",
+    c("predictions", "residuals","predictions","observations"))
+  )
> #
> # Diagnostic variogram: lack of residual correlation
> #
> make.gmCompositionalGaussianSpatialModel(
+   data=xv.res.comp, coords = wind.coords, V="alr") %>%
+   variogram(cutoff=50, width=5, alpha=90*1:2) %>% plot
> #
> # Diagnostic maps: lack of residual spatial structure
> #
```

```
> pairsmap(data=xv.res.comp, loc=wind.coords, mfrow=c(6,1),
+        cexrange=c(1,1)*1.5, foregroundcolor=NA)
> #
> # 2.- Cross-validation with a separate validation set
> #
> # split data in training and validation subsets
> windarlingS = windarling %>%
+   dplyr::filter(Sample.West+Sample.East==1)
> windarling.subset = with(windarling, Sample.West+Sample.East==1)
> windS.coords = wind.coords[windarling.subset,]
> windS.compo = wind.compo[windarling.subset,]
> windarlingJ = windarling %>%
+   dplyr::filter(Sample.West+Sample.East==0)
> windJ.coords = wind.coords[!windarling.subset,]
> windJ.compo = wind.compo[!windarling.subset,]
> # set spatial model
> wind.ng = KrigingNeighbourhood(nmax=20, nmin=4, maxdist=20)
> windS.alr.ggAnis = make.gmCompositionalGaussianSpatialModel(
+   data=acomp(windS.compo), coord=windS.coords, V="alr",
+   ng = wind.ng, formula=~1, model = wind.pwlr.lmcAnis)
> # prediction
> windJ.xv = predict(windS.alr.ggAnis,
+                    newdata=data.frame(windJ.coords))
> # validation
> windJ.xv.true = data.frame(alr(windJ.compo))
> windJ.xv.preds = windJ.xv[,grep("pred", colnames(windJ.xv))]
> windJ.xv.vars = windJ.xv[,grep("var", colnames(windJ.xv))]
> windJ.xv.res = windJ.xv.preds-windJ.xv.true
> colnames(windJ.xv.res)=c("alr1.residuals","alr2.residuals",
+                          "alr2.residuals","alr4.residuals",
+                          "alr5.residuals")
> windJ.xv.zscore=windJ.xv.res/sqrt(windJ.xv.vars)
> #
> # Diagnostic plots
> #
> par(mfcol=c(5,5), mar=c(4,4,3,1))
> for(i in 1:5){
+   # observed vs predicted
+   plot(windJ.xv.true[,i]~windJ.xv.preds[,i],ylab="observed",
+        xlab="predicted", asp=1, main=paste("alr",i,sep=""))
+   abline(a=0, b=1)
+   abline(lm(windJ.xv.true[,i]~windJ.xv.preds[,i]), col=2)
+   # residual histogram
+   hist(windJ.xv.res[,i], xlab="residual",main="")
+   # residual QQ plot
+   qqnorm(windJ.xv.res[,i], main="normal QQ plot")
+   qqline(windJ.xv.res[,i], col=2)
+   # standardised residuals vs predictions
+   plot(windJ.xv.zscore[,i]~windJ.xv.preds[,i],
+        ylab="standardised errors", xlab="predicted")
+   abline(h=0)
+   abline(lm(windJ.xv.zscore[,i]~windJ.xv.preds[,i]), col=2)
+   # residuals vs covariables
+   boxplot((windJ.xv.res)[,i]~windarlingJ$Lithotype,
```

```
+            horizontal=T)
+ }
> # multivariate error measures
> xvErrorMeasures(windJ.xv, observed=windJ.xv.true,
+                 output=c("MSDR1","MSDR2","MSE"))
> windJ.mahNorms2 = xvErrorMeasures(windJ.xv,
+                  observed=windJ.xv.true, output="Mahalanobis")
> hist(windJ.mahNorms2, breaks=100)
> qqplot(windJ.mahNorms2, rchisq(5000, df=5),
+        xlab="observed quantiles", ylab="chi2(df=5) quantiles")
> abline(a=0, b=1, col=2)
> # Accuracy and precision of the model
> wind.compo.xvacc= accuracy(windJ.xv, observed=windJ.xv.true,
+                            method = "cokriging" )
> mean(wind.compo.xvacc)
> precision(wind.compo.xvacc)
```

B.3 Estimation

Once a model for the spatial continuity has been derived, estimation can be performed, if that is the goal of the study.

The steps for this are as follows. Having decided on a suitable LMC and hence on an estimation method,

1. Use cross-validation to determine the most appropriate neighbourhood parameters for the estimation (combine `KrigingNeighbourhood` with `make.gmCompositionalGaussianSpatialModel`; convert with `as.gstat`, validate with `gstat.cv`)
2. Define an estimation grid (`GridTopology`, `SpatialGrid`) and if necessary, constrain the estimation region to avoid extrapolation to regions without samples. (`constructMask`, then `setMask`)
3. Calculate Cokriging estimates on the estimation grid (`predict`)
4. If using a masked grid, unmask it (`unmask`)
5. Preliminary check that the results are reasonable by means of spatial maps (`pairsmap`), summary statistics, density plots (`density`), QQ-plots (`qqplot`) and ternary diagrams (`plot.acomp`).

The code for these calculations makes use of "`compositions`", "`gstat`" and "`gmGeostats`". Detailed explanations can be found in Chap. 6.

```
> #
> # Cross-validation to assess neighbourhood settings:
> # experiment with nmax, nmin and maxdist
> #
> wind.alr.xvAnis= gstat.cv(wind.alr.ggAnis,remove.all=TRUE,
+                           all.residuals=TRUE,verbose=FALSE,
+                           nmax=20, nmin=4, maxdist=20)
> wind.ng = KrigingNeighbourhood(nmax=20, nmin=4, maxdist=20)
```

```
> #
> # Grid definition and masking
> #
> xmin=floor(min(wind.coords[,1]))
> xmax=ceiling(max(wind.coords[,1]))
> ymin=floor(min(wind.coords[,2]))
> ymax=ceiling(max(wind.coords[,2]))
> x0 = c(xmin, ymin)
> names(x0) = colnames(wind.coords)
> Dx = c(1,1)
> nx = c(xmax-xmin, ymax-ymin)/Dx +1
> wind.gt = GridTopology(x0, Dx, nx)
> wind.grid.fine = SpatialGrid(wind.gt)
> wind.mask.fine =
+   constructMask(wind.grid.fine, method="maxdist", maxval=7,
+                              x=cbind(wind.coords,wind.compo) )
> wind.grid.fine.masked = setMask(wind.grid.fine, wind.mask.fine)
> #
> # Ordinary Cokriging: spatial object, prediction, unmasking
> #
> wind.alr.ggAnis = make.gmCompositionalGaussianSpatialModel(
+  data=wind.compo, coords = wind.coords, V="alr", ng = wind.ng,
+  formula = ~1, model=wind.pwlr.mdAnis)
> wind.alr.ock = wind.alr.ggAnis %>%
+   predict(newdata=wind.grid.fine.masked) %>%
+   unmask(wind.mask.fine)
> # recasting to compositions
> wind.compo.ock = gsi.gstatCokriging2compo(wind.alr.ock,
+                        V="alr", orignames=colnames(wind.compo))
> #
> # Universal Co-kriging: isotropic variogram model
> #
> aux = wind.alr.ggAnis %>% as.gstat
> for (i in 1:length(aux$model)){
+   aux$model[[i]]$anis1=1}
> wind.pwlr.md = as.LMCAnisCompo(aux$model, V="alr")
> # geospatial object
> wind.alr.gg.uck = make.gmCompositionalGaussianSpatialModel(
+   data=wind.acomp, coords = wind.coords, V="alr",
+   ng=wind.ng, formula = ~1+Northing, model = wind.pwlr.md)
> # prediction, unmasking, recasting to compositions
> wind.alr.uck =
+   predict(wind.alr.gg.uck, newdata=wind.grid.fine.masked) %>%
+         unmask(wind.mask.fine)
> wind.compo.uck = gsi.gstatCokriging2compo(wind.alr.uck,
+                        V="alr", orignames=colnames(wind.compo))
> #
> # Check of consistency between estimates and samples
> #
> # Spatial maps
> dsc=image_cokriged(wind.compo.ock, ivar="Fe")
> points(wind.coords, cex=0.5, col="black", pch=22,
+        bg=dsc$col[cut(wind.acomp[,"Fe"], breaks=dsc$breaks)])
> # Summary statistics
```

```
> summary(wind.compo.ock[wind.mask.fine,(1:6)])
> summary(wind.acomp[,1:6])
> # Density plots
> par(mfrow=c(2,3))
> for (i in colnames(wind.compo)){
+   kde = list()
+   kde[["ock"]] = density(100*wind.compo.ock[wind.mask.fine,i])
+   kde[["data"]] = density(wind.compo[,i])
+   xlim = range(sapply(kde, function(x) x$x))
+   ylim = range(sapply(kde, function(x) x$y))
+   plot(kde[["data"]], col="black", xlim=xlim, ylim=ylim, main=i)
+   lines(kde[["ock"]], col="red")
+ }
> # QQ-plots
> par(mfrow=c(3,2))
> for (i in 1:6){
+ x=quantile(wind.compo[,i],probs = seq(0,1,0.01))
+ y1=quantile(wind.compo.ock[wind.mask.fine,i]*100,
+             probs = seq(0,1,0.01),na.rm=T)
+ plot(x,y1,xlab="true",ylab="OCK-predicted",asp=1)
+ abline(a=0,b=1)
+ }
> # Biplots
> par(mfrow=c(1,1))
> OCK.pc = wind.compo.ock[wind.mask.fine,] %>% clr %>% princomp
> coloredBiplot(OCK.pc, choices = 1:2, xlabs.pc=1, cex=c(0.25,1))
> # Ternary diagrams
> par(mfcol=c(1,2), mar=c(4,4,2,4))
> subcompo1 =c("Mn", "Al2O3", "SiO2")
> subcompo2 = c("P", "R", "Fe")
> wind.compo.ock[,subcompo1] %>% na.omit %>% acomp %>%
+   plot(center=T, pca=T, col.pca=2, pch=".")
> title(main="OCK")
> wind.compo.ock[,subcompo2] %>% na.omit %>% acomp %>%
+   plot(center=T, pca=T, col.pca=2, pch=".")
```

B.4 Simulation

If the objective of the study is to generate equiprobable realisations of the compositional random function, then the workflow steps of numerical and spatial EDA remain applicable. The workflow subsequent to these steps depends on whether the simulation algorithm is based on two point or multiple point statistics. Specifically,

1. For a simulation algorithm based on two point statistics
   - If the algorithm assumes multivariate normality, check if the condition is satisfied, otherwise use the Flow Anamorphosis (FA) introduced in Chap. 8 to convert the data to multivariate standard normal data (`ana`).

- Compute experimental variograms for the FA transformed data, fit a suitable LMC to them and validate the model, alternatively use MAF to spatially decorrelate the FA data, determine and validate the model of spatial continuity of the MAF factors (`Maf`).
- Assess accuracy, precision and goodness of the spatial model (`accuracy`, `mean`, `prediction`).
- If goodness of the spatial model is acceptable, calculate realisations of the data, either MAF or FA (`SequentialSimulation` for specifying parameters; `predict` for doing calculations).
- Back-transform to raw compositions (`gmApply`, with eventually `ana` inverted, `predict` for maf scores, `ilrInv` or `alrInv` for recovering compositions).

2. For an algorithm based on multiple point statistics

- Preprocess data as required for the algorithm (`SpatialPixelsDataFrame`, `make.gmCompositionalMPSSpatialModel`).
- Define a grid (`GridTopology`, `SpatialGrid`), and eventually a mask (`constructMask`, `setMask`).
- Specify algorithm parameters (`DSpars`).
- Execute the simulation algorithms to obtain equiprobable realisations (`predict`).
- Back-transform to raw compositions (if needed, current direct sampling algorithm returns compositions directly).

The code for these calculations makes use of "`compositions`", "`gstat`" and "`gmGeostats`". Detailed explanations can be found in Chaps. 9 and 10.
This is the workflow for sequential Gaussian simulation.

```
> #
> # Workflow for sequential Gaussian co-simulation
> #
> #
> # Transformation to normal scores via FA, note sigma0 and
> # sigma1 need to be determined experimentally for each dataset
> #
> fana.wind = ana(wind.ilr, sigma0=0.2 , sigma1=1.2)
> wind.ns = fana.wind(wind.ilr)
> # Normality checks
> plot(data.frame(wind.ns), panel=vp.kde2dplot,
+      colpalette = spectralcolors)
> par(mfcol=c(2,5), mar=c(3,3,2,1))
> for(i in 1:5){
+   hist(wind.ns[,i], main=colnames(wind.ns)[i])
+   qqnorm(wind.ns[,i], main="")
+   qqline(wind.ns[,i], col=2)
+ }
> library(MVN)
> mvn(wind.ns,mvnTest=c("mardia"),univariateTest="SW",desc=FALSE)
> mvn(wind.ns,mvnTest=c("hz"),univariateTest="Lillie",desc=FALSE)
> mvn(wind.ns,mvnTest=c("energy"),univariateTest="AD")
```

```
> #
> # Modelling the spatial continuity of the FA transformed data
> #
> wind.ns.sdf = data.frame(wind.coords, wind.ns) # spatial data
> wind.ns.gg = NULL  # empty object
> for(i in 1:5){ # loop to add gstat layers
+   id = colnames(wind.ns)[i]
+   frm = paste(id, "~1", sep="")
+   wind.ns.gg = gstat(g=wind.ns.gg, id=id, data=wind.ns.sdf,
+          formula=as.formula(frm), locations=~Easting+Northing,
+          nmax=20, nmin=4, maxdist=20)
+ }
> # alternative: use make* statt gstat:
> wind.ns.gg = make.gmMultivariateGaussianSpatialModel(
+   data = data.frame(wind.ns), coords = wind.coords,
+   nmax=20, nmin=4, maxdist=20)
> # empirical variogram
> wind.ns.vgAnis=variogram(wind.ns.gg,cutoff=40, width=5,
+                          alpha=c(90,180),tol.hor=45)
> # variogram model
> wind.ns.mdAnis = vgm(nugget=0.1, psill =0.3, range=15,
+                                 model = "Sph", anis=c(90, 0.5))
> wind.ns.mdAnis = vgm(add.to=wind.ns.mdAnis, psill =0.6,
+                      range=50, model = "Sph", anis=c(90, 0.5))
> # variogram mode fit // extension of the gstat object
> wind.ns.gg = fit_lmc(v=wind.ns.vgAnis, g=wind.ns.gg,
+                      model=wind.ns.mdAnis, fit_lmc=TRUE,
+                      correct.diagonal = 1.001)
> variogramModelPlot(wind.ns.vgAnis,wind.ns.gg$model)
> #
> # Cross-validation and accuracy and precision calculations for
> # the model of the FA scores
> # (follow the workflow in Sect. B.2)
> #
> # Construct simulation grid and mask if required
> #
> xmin=floor(min(wind.coords[,1]))
> xmax=ceiling(max(wind.coords[,1]))
> ymin=floor(min(wind.coords[,2]))
> ymax=ceiling(max(wind.coords[,2]))
> xx = seq(from=xmin, to=xmax, length.out=xmax-xmin+1)
> yy = seq(from=ymin, to=ymax, length.out=ymax-ymin+1)
> # option 1: with expand.grid
> wind.grid.fine = expand.grid(x=xx, y=yy)
> colnames(wind.grid.fine) = colnames(wind.coords)
> # option 2: use GridTopology
> x0 = c(xmin, ymin)
> names(x0) = colnames(wind.coords)
> Dx = c(1,1)
> nx = c(xmax-xmin, ymax-ymin)/Dx +1
> wind.gt = GridTopology(x0, Dx, nx)
> wind.grid.fine = SpatialGrid(wind.gt)
> #
> # Masking
```

```
> #
> wind.mask.fine =
+   constructMask(wind.grid.fine, method="maxdist", maxval=7,
+                                 x=cbind(wind.coords,wind.compo) )
> wind.grid.fine.masked = setMask(wind.grid.fine, wind.mask.fine)
> #
> # Perform simulation
> #
> wind.grid.fine.masked = setMask(wind.grid.fine,
+                                 mask = wind.mask.fine)
> wind.ns.cosim.aux = predict(wind.ns.gg,
+                   newdata=wind.grid.fine.masked,
+                   nsim=100, debug.level = -1)
> # alternative: using make* statt gstat:
> wind.sgs = SequentialSimulation(nsim = 100, debug.level = -1,
+                                 ng=wind.ng)
> wind.ns.cosim.aux = predict(wind.ns.gg, pars=wind.sgs,
+                   newdata=wind.grid.fine.masked)
> #
> # cast to DataFrameStack
> #
> # choose one!
> dt = wind.ns.cosim.aux[,-(1:2)] # if expand.grid was used
> dt = wind.ns.cosim.aux@data # if GridTopology was used
> # recast!
> wind.ns.cosim = dt %>% DataFrameStack(dimnames=list(
+     loc=1:sum(wind.mask.fine), sim=1:100,
+     var=colnames(wind.ns)), stackDim="sim")
> #
> # Backtransform first to ilr scores, then to compositions
> #
> # compute the inverse anamorphosis
> wind.ilr.cosim.aux = gmApply(X=wind.ns.cosim,
+                           FUN=fana.wind, inv=T)
> # compute the inverse ilr
> wind.compo.cosim.aux = gmApply(X=wind.ilr.cosim.aux,
+                           FUN=ilrInv, orig=wind.acomp)
> # unmask
> wind.compo.cosim = unmask(wind.compo.cosim.aux,
+                         mask=wind.mask.fine,
+                         fullgrid = wind.grid.fine )
```

This is the workflow for MPS direct simulation.

```
> #
> # Workflow for DSSIM (assuming that the objective is to
> # simulate the spatial distribution in a region using
> # information from a more densely sampled adjacent region)
> #
> windDS.subset = (windarling$West+windarling$Sample.East)==1
> windDS.gcoords = wind.coords[windDS.subset,]
> windDS.gcompo = wind.compo[windDS.subset,]
> plot(windDS.gcoords, asp=1)
> # Definition of conditioning data
> wind.compo.cond = windDS.gcompo[windDS.gcoords$Easting>=0,]
```

```
> wind.coords.cond = windDS.gcoords[windDS.gcoords$Easting>=0,]
> #
> # Definition of simulation grid
> #
> aux = get.knn(windDS.gcoords, k=1, algo="kd_tree")$nn.dist
> cellSize = floor(median(aux))
> mmin=min(windDS.gcoords[,1])
> mmax=max(windDS.gcoords[,1])
> x = seq(mmin-cellSize/2,mmax+cellSize/2,cellSize)
> mmin=min(windDS.gcoords[,2])
> mmax=max(windDS.gcoords[,2])
> y = seq(mmin-cellSize/2,mmax+cellSize/2,cellSize)
> nx = length(x)
> ny = length(y)
> #
> # establish grid
> #
> windDS.ggrid = GridTopology(cellsize=c(cellSize, cellSize),
+   cellcentre.offset=c(min(x), min(y)), cells.dim=c(nx,ny))
> names(windDS.ggrid@cellcentre.offset)=colnames(windDS.gcoords)
> #
> # Migration of data to grid
> #
> windDS.gcgrid = windDS.gcoords %>% SpatialPoints %>%
+  SpatialPixelsDataFrame(data=windDS.gcompo, grid=windDS.ggrid)
> #
> # Definition of training image
> #
> tk.TI = coordinates(windDS.gcgrid)[,"Easting"]<0
> wind.TI0 = windDS.gcgrid[tk.TI,]
> windDS.cgrid0 =  windDS.gcgrid[!tk.TI,]
> #
> # Mirror TI to simulation region
> #
> windDS.grid = windDS.ggrid
> windDS.grid@cellcentre.offset[1]=min(x[x>0])
> windDS.grid@cells.dim[1] = sum(x>0)
> windDS.cgrid = windDS.cgrid0 %>% as("SpatialPixels") %>%
+   SpatialPixelsDataFrame(windDS.cgrid0@data, grid=windDS.grid)
> wind.TI = flipHorizontal(as(wind.TI0,"SpatialGridDataFrame"))
> #
> # define a mask to constrain the simulation region
> #
> windDS.mask=constructMask(grid=windDS.grid, method="maxdist",
+                  maxval = 10, x = as.data.frame(windDS.cgrid))
> #
> # Create spatial object, set parameters and simulate
> #
> windDS.cmodel = make.gmCompositionalMPSSpatialModel(
+              data = windDS.cgrid, V = "ilr", model = wind.TI )
> windDS.pars =
+   DSpars(nsim=25, scanFraction=0.75, patternSize=6, gof=0.05)
> wind.dsim =  predict(windDS.cmodel, pars=windDS.pars,
+                 newdata=setMask(windDS.grid, mask=windDS.mask))
```

B.5 Evaluation and Postprocessing

Having generated either estimates or simulated realisations, the results are postprocessed to provide summary outputs. We start with statistics which are of greater relevance to simulation output than estimation results. For simulation as a first step, the following are generated:

1. Maps of individual realisations (`getStackElement`, `image_cokriged`).
2. Statistical maps such as E-type estimates and an estimate of the total variation of the realisations (`gmApply` e.g. combined with `mean`; `image_cokriged`).
3. Checks of the reproduction of target marginal and two point statistics, including boxplots of compositional means, swarms of QQ-plots to check the shapes of the realisation distributions and related numerical measures and variogram swarms (`gmApply`, `boxplot`, `swarmPlot`).
4. Selectivity curves for the composition.

For estimation, the reproduction of target statistics is also of importance as is the generation of selectivity curves.

The code for these calculations makes use of "`compositions`" and "`gmGeostats`". Detailed explanations can be found in Chap. 11.

```
> #
> # Maps of individual realisations
> #
> out = wind.grid.fine %>%
+     cbind(getStackElement(wind.compo.cosim,1)) %>%
+     image_cokriged(ivar="Fe", legendPos = "top")
> points(wind.coords, pch=21, col=1, cex=0.75,
+          bg=out$col[cut(wind.acomp[,"Fe"],out$breaks)])
> # alternative: if class(wind.compo.cosim)=="Spatial*"
> out = getStackElement(wind.compo.cosim,1) %>%
+     image_cokriged(ivar="Fe", legendPos = "top")
> points(wind.coords, ...)
> #
> # Calculation and plots of statistical maps
> #
> wind.cosim.Etype = wind.compo.cosim %>%
+   gmApply(c("loc","var"), geometricmean) %>% clo
> wind.sim.Etype = wind.compo.sim %>%
+   gmApply(c("loc","var"), geometricmean) %>% clo
> wind.cosim.totvar = wind.compo.cosim %>%
+   gmApply("loc", function(x) mvar(acomp(x)) )
> aux = cbind(wind.grid.fine, wind.cosim.Etype)
> myfun = function(j){
+   out = image_cokriged(aux, ivar=j, legendPropSpace = 0.2,
+                        legendPos = "top")
+   points(wind.coords, pch=21, col=1, cex=0.5,
+          bg=out$col[cut(wind.acomp[,j],out$breaks)])
+ }
> aux = cbind(wind.grid.fine, totvar=wind.cosim.totvar)
> image_cokriged(aux, ivar="totvar", legendPos = "top")
```

```
> points(wind.coords, pch=21, col=1, cex=0.25)
> #
> # Boxplots of compositional means
> #
> a =  clo(apply(wind.compo.cosim, MARGIN=c("sim","var"),
+                FUN= geometricmean, na.rm=T))
> boxplot(data.frame(a), log="y", border=2)
> points(1:6, mean(wind.acomp), lty=2)
> #
> # Swarms of QQ plots
> #
> myQQfun = function(x, j="Fe"){
+   erg = qqplot(wind.acomp[,j], x[,j], plot.it = F)
+   return(data.frame(erg))
+ }
> auxfun = function(j){
+   a = swarmPlot(wind.compo.cosim.aux, PLOTFUN = myQQfun, j=j,
+            .plotargs = list(asp=1, xlab="observed quantiles",
+            ylab="simulation quantiles", main=j) )
+   abline(a=0, b=1, col=2)
+   return(a)
+ }
> a = myQQfun(getStackElement(wind.compo.cosim.aux,1))
> auxFe = auxfun(j="Fe")
> #
> # Numerical summaries of deviation between target and
> # realisation quantiles
> #
> aa = lapply(colnames(wind.compo), auxfun)
> meanSqDev = function(x) mean((x$x-x$y)^2)
> res = lapply(aa, function(aaa) sapply(aaa, meanSqDev))
> names(res) = colnames(wind.compo)
> boxplot(data.frame(res))
> #
> # Variogram swarms
> #
> vgFe =
+   gstat(id="Fe", formula=Fe~1, locations=~Easting+Northing,
+         data = cbind(data.frame(wind.acomp), wind.coords)) %>%
+   variogram(cutoff=50)
> myVGfun = function(x){
+ vg = gstat(id="Fe", formula=Fe~1, locations=~Easting+Northing,
+        data = cbind(data.frame(x), wind.grid.fine.masked)) %>%
+                 variogram(cutoff=50)
+ return(data.frame(x=c(0,vg$dist),y=c(0,vg$gamma)))
+ }
> aux = swarmPlot(wind.compo.cosim.aux, PLOTFUN= myVGfun,
+                  .plotargs=F)
> plot(aux, xlim=range(0, vgFe$dist), ylim=range(0, vgFe$gamma),
+      xlab="lag distance", ylab="semivariogram")
> points(vgFe$dist, vgFe$gamma, col=2)
> #
> # Selectivity curves
> #
```

```
> TonnageCurve = function(X, vr=1, cutoffs=
+       seq(from=min(X[,vr]), to=max(X[,vr]), length.out=201)){
+   t = rowMeans(outer(cutoffs, X[,vr], "<="))
+   return(data.frame(cutoff=cutoffs,tonnage=t))
+ }
> GradeCurve = function(X, vr=1, cutoffs=
+       seq(from=min(X[,vr]), to=max(X[,vr]), length.out=201)){
+   z = X[,vr]
+   t = rowMeans(outer(cutoffs, z, "<="))
+   g = outer(cutoffs, z, "<=") %*% z
+   return(data.frame(cutoff=cutoffs,grade=g/(t*length(z))))
+ }
> # tonnage:
> aux = swarmPlot(wind.compo.cosim.aux, PLOTFUN=TonnageCurve,
+                 vr="Fe", .plotargs = F)
> plot(aux, xlab="cutoff", ylab="tonnage", main="Fe")
> aux0 = TonnageCurve(wind.acomp, vr="Fe")
> lines(aux0[,1], aux0[,2], col=2)
> # grade
> aux = swarmPlot(wind.compo.cosim.aux, PLOTFUN=GradeCurve,
+                 vr="Fe", .plotargs = F)
> plot(aux, xlab="cutoff", ylab="grade", main="Fe")
> aux0 = GradeCurve(wind.acomp, vr="Fe")
> lines(aux0[,1], aux0[,2], col=2)
```

Further postproceesing includes change of support by reblocking (`aggregate`) and an example is shown below.

```
> #
> # Input data: E-type spatial points data frame
> #
> spMean = SpatialPointsDataFrame(coords=wind.grid.fine,
+                                 data=data.frame(wind.sim.Etype))
> #
> # construct the block grid topology
> #
> (rgs = gmApply(coordinates(wind.grid.fine),2,range))  # range
> apply(rgs,2,diff)/5  # nr of blocks for size 5m
> wind.blockgrid.topo = GridTopology( cells.dim = c(89, 22),
+     cellcentre.offset = c(-240,15), cellsize = c(5,5)  )
> # block spatial grid
> wind.blockgrid = SpatialGrid(wind.blockgrid.topo)
> #
> # Aggregation, additive upscaling
> #
> spAggr = aggregate(spMean, by=wind.blockgrid, FUN=mean)
> # display
> cbind(coordinates(spAggr), spAggr@data) %>%
+     image_cokriged(ivar="Fe")
> myAggrFun = function(x){
+   spx = SpatialPointsDataFrame(coords=wind.grid.fine,
+                                data=data.frame(x))
+   spAggr = aggregate(spx, by=wind.blockgrid, FUN=mean)
+   return(spAggr@data)
```

```
+ }
> wind.compo.sim.block = gmApply(wind.compo.sim, FUN=myAggrFun)
> dim(wind.compo.sim.block)
> dimnames(wind.compo.sim.block)[-1]
> # Visualisation
> myfun = function(i,j){
+   bks = quantile(wind.acomp[,j], prob=(0:10)/10)
+   out = coordinates(gtB) %>%
+   cbind(getStackElement(wind.compo.sim.block,i)) %>%
+     image_cokriged(ivar=j, legendPos = "top", breaks = bks)
+   points(wind.coords, pch=21, col=1, cex=0.75,
+     bg=out$col[cut(wind.acomp[,j],out$breaks)])
+ }
> manipulate(myfun(i,j),
+            i=slider(1, max = 100),
+            j=picker("Fe", "Al2O3", "SiO2", "Mn", "P", "R"))
```

Index

R. Tolosana-Delgado, U. Mueller, *Geostatistics for Compositional Data with R*,
Use R!, https://doi.org/10.1007/978-3-030-82568-3