TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Exploratory Tools for Cellwise Outlier Detection in Compositional Data with Structural Zeros

Ausgeführt am

Institut für Stochastik und Wirtschaftsmathematik

unter der Leitung von

Privatdoz. Dipl.-Ing. Dr.techn. Matthias Templ

durch

Lukas Beisteiner, BSc

Augasse 1, 2811 Wiesmath

Abstract

The analysis of compositional data using the log-ratio approach is based on ratios between the compositional parts. Zeros in the parts thus cause severe difficulties for the analysis. Log-ratio transformations represent the compositional information into new coordinates. Outliers within these coordinates may be detected, however it remains unclear which particular parts of the composition led to the deviating ratios in question. To address this issue, the thesis presents four exploratory tools for identifying cellwise outliers in compositional data sets with structural zeros. In order to deal with structural zeros the proposed methods use robust imputation methods or split the data into subcompositions determined by their zero patterns. Ratios between parts are analyzed using an isometric log-ratio transformation or by observing pairwise log-ratios. Combining the results from robust regression analysis and robust distance calculations the approaches deduce row- and cellwise outliers within the original sample space. All four methods are applied on the household expenditure data from Albania and then compared. A close-to-reality simulation study is conducted to assess the performance of the different outlier detection algorithms.

Acknowledgments

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich im Zuge meiner Diplomarbeit und meines Studiums unterstützt haben. Zuerst möchte ich mich bei meinem Professor Matthias Templ bedanken. Du standest mir fachlich und inhaltlich stets zur Seite und hast mir sehr viel deiner wertvollen Zeit geschenkt. Vielen Dank für die großartige Zusammenarbeit.

Ein besonderer Dank gilt meinen Freunden. Danke, dass ihr mich während dieses Studiums begleitet habt, dass ihr immer ein offenes Ohr für mich habt und dass man mit euch so viel erleben kann. Ich hoffe ich bin euch ein ebenso guter Freund.

Zu guter Letzt möchte ich mich auch noch bei meinen Schwester und meinen Eltern bedanken. Ich bin sehr froh zu wissen, dass ihr, egal welche Entscheidungen ich treffe und welche Ziele ich mir setze, immer hinter mir steht und mich stets unterstützt.

Contents

Chapter 1

Introduction

The term compositional data is used when relative contributions of parts on a whole contain the information of interest. Such data occur frequently in many practical situations. A typical example would be household expenditures on various costs like housing, foodstuff, alcohol and tobacco, furnishings, health and transportation for a sample of households, whenever the aim is to analyze the ratios between parts or/and the multivariate structure of the data. Such data from world bank will be discussed and analyzed later in this thesis. The sum of these parts is not necessarily 1 (or 100%), but since the relevant information is contained in the ratios between the parts, a constant sum constraint 1 or 100 can be achieved without any loss of information.

Compositional data induce an own sample space; they are represented in the simplex, correspondent to the Aitchison geometry (see Egozcue et al. [2003] and Bacon-Shone [2008]) that is substantially different from the Euclidean geometry. The D-part simplex is defined as

$$S^D = \left\{ \boldsymbol{x} = (x_1, \ldots, x_D) \mid x_i > 0, i = 1, \ldots, D; \sum_{i=1}^{D} = \kappa \right\}$$

where κ is an arbitrary positive constant. Therefore, standard statistical methods designed for the Euclidean geometry cannot be directly applied to compositions. To offer a way out Aitchison [1986] proposed a family of log-ratio coordinates that enable to express compositional data from the simplex in the Euclidean real space. Nowadays, the isometric log-ratio (ilr) coordinates are preferred due to advantageous theoretical properties like isometry and non-singularity. We will also make extensive use of this transformation within this thesis. The downside comes with the fact that the new and transformed variables are not easy to interpret. In this thesis we aim for detecting outliers in the data only. Therefore the interpretability of the
variables/coordinates becomes a non-issue. As stated above after transformation, standard statistical methods and in this sense outlier detection methods can now be applied on the variables represented in the new coordinates. As a consequence all outlier results are expressed in the transformed space as well. However from a user's perspective it may be of great interest which of the initial part of an observation leads

to deviating ratios within the transformed coordinates.

This thesis therefore proposes several exploratory tools for detecting cellwise outliers within compositional data. Chapter 2 elaborates on the theoretical background of the data transformation and robust methods used for outlier detection. Chapter 3 then describes the proposed algorithms for a cellwise outlier detection in detail, also introducing two different approaches for dealing with structural zeros in the data. The methods are then applied on household expenditure data from Albania and the obtained results are discussed in Section 4. The thesis concludes in Chapter 5 by measuring the performance of the different outlier detection algorithms by means of a simulation study, based on the original data from the survey.

Chapter 2

Basic notations and definitions

This chapter will first recapitulate robust properties of an estimator, most important the *breakdown point*, *efficiency* and *affine equivariance*. We will introduce an *isometric log-ratio transformation* used for representing the compositional variables into orthogonal coordinates in the Euclidean real space. Furthermore the theory behind outlier detection using *robust regression*, as well as *univariate and multivariate distances* is presented.

With real-world data sets with quantitative information, estimators which are not sensitive to outliers should be preferred. Therefore a measure for the sensitivity of an estimator to contaminated data has to be established. In this case the *finite-sample breakdown point* ("Donoho-Huber breakdown point") (see Donoho and Huber [1983]) is used. In a given sample $x_1, \dots, x_n$ replace m data points $x_{1_1}, \dots, x_{1_m}$ with arbitrary values $y_1, \dots, y_m$. Denote these new data points as $z_1, \dots, z_n$. The (gross-error) finite-sample breakdown point for estimator T is then defined as

$$\varepsilon_n^*(T; x_1, \dots, x_n) = \min \left\{ \frac{m}{n}; \max_{i_1,\dots,i_m} \sup_{y_1,\dots,y_m} |T(z_1, \dots, z_n)| = \infty \right\}.$$

In general ε_n^* is independent of $x_1, \dots, x_n$. Furthermore Hampels (see Hampel [1971]) asymptotic version of this definition for infinite samples results in

$$\lim_{n \to \infty} \varepsilon_n^* = \varepsilon^*.$$

The maximum achievable breakdown point is 50%. A breakdown point over 50% is not feasible; in this case the majority of outliers could be treated as "good" data points.

Robust estimates are more reliable in case of contaminated or extreme data points in a data set, however they do have a drawback when it comes to the quality of the estimate. Contrary to their classical counterpart, robust estimates are in general not very efficient. Concerning the precision of an estimator, statistical efficiency is an important and desirable characteristic. It is defined via the *Ramér-Crao bound* or *Cramér-Rao inequality*, which is a lower bound for the variance of an estimator. Let $x_1, \dots, x_n$ be a sample from a distribution $\boldsymbol{X}$, for which the density $f(x; \theta)$ exists with unknown parameter $\theta \in \Theta$. Additionally we consider the following regularity assumptions:

1. **(R1)** The true parameter θ_0 is an inner point of Θ and from $\theta \neq \theta'$ follows $f(x;\theta) \neq f(x;\theta')$.

2. **(R2)** The distributions $f(x_i;\theta), i = 1,\dots,n$, have common support for all $\theta \in \Theta$.

3. **(R3)** $f(x;\theta)$ is twice differentiable in θ and for $k = 1, 2$ holds

$$\frac{\partial^k}{\partial\theta^k}\int_{-\infty}^{\infty} f(x;\theta)dx = \int_{-\infty}^{\infty} \frac{\partial^k f(x;\theta)}{\partial\theta^k}dx.$$

   Thus the operations of integration with respect to x and differentiation with respect to θ can be permuted.

The regularity assumptions above are necessary for the Cramér-Rao bound and are not only sufficient for the existence of the *Fisher information* $I(\theta)$, defined by

$$I(\theta) = \mathbb{E}\left[\left(\frac{\partial \ln f(X;\theta)}{\partial\theta}\right)^2\right].$$

The *Cramér-Rao inequality* can then be introduced as follows: Consider a sample $x_1,\dots,x_n$ from a distribution X, for which the density $f(x;\theta)$ exists with unknown parameter $\theta \in \Theta$. Let $T = T(x_1,\dots,x_n)$ be an estimate for the parameter θ with $\mathbb{E}[T(x_1,\dots,x_n)] = k(\theta)$. Given the regularity assumptions (R1)-(R3) the following inequality holds true

$$\mathrm{Var}[T] \geq \frac{[k'(\theta)]^2}{nI(\theta)}. \tag{2.1}$$

For an *unbiased* estimate T for θ, meaning that $\mathbb{E}[T] = \theta$, the inequality reduces to

$$\mathrm{Var}[T] \geq \frac{1}{nI(\theta)}.$$

The *efficiency* $e(T)$ of an unbiased estimator T is defined by the quotient of the Cramér-Rao bound and the true variance of the estimator:

$$e(T) = \frac{\frac{1}{nI(\theta)}}{\mathrm{Var}[T]}$$

An unbiased estimate is called *efficient* if its variance reaches the Cramér-Rao bound, which implies that $e(T) = 1$ for efficient estimates. If the variance of an unbiased estimate reaches the Cramér-Rao bound only for $n \to \infty$ the estimate is called *asymp-*

totically efficient:

$$\lim_{n\to\infty} \frac{\frac{1}{nI(\theta)}}{\mathrm{Var}[T]} = 1$$

In case of robust estimators, most of them are *biased*, $b(\theta) = \mathbb{E}(T) - \theta$, and thus $k(\theta) = b(\theta) + \theta$. By the result in Formula (2.1), any unbiased estimator whose expectation is $k(\theta)$ has variance greater than or equal to $[k'(\theta)]^2/(nI(\theta))$. Therefore any estimator T whose bias is given by function $b(\theta)$ satisfies

$$\mathrm{Var}[T] \geq \frac{[1 + b'(\theta)]^2}{nI(\theta)}. \tag{2.2}$$

The unbiased version of the bound is a special case of this result, with $b(\theta) = 0$. From Formula (2.2) we find that the *mean squared error* of a biased estimator, given by

$$\mathrm{MSE}(T) = \mathbb{E}[(T - \theta)^2] = \mathrm{Var}[T] + b(T)^2, \tag{2.3}$$

is bounded by

$$\mathbb{E}[(T - \theta)^2] \geq \frac{[1 + b'(\theta)]^2}{nI(\theta)} + b(\theta)^2,$$

using the standard decomposition of the MSE as in Formula (2.3). It is important to note that this bound can be less than the unbiased Cramér-Rao bound bound $1/(nI(\theta))$. This may be the case when using robust estimators, which are biased but achieve lower variance, leading to a low mean squared error.

Another desirable characteristic of an estimator is the so called *affine equivariance*. This property is specifically interesting when it comes to multivariate estimates of location and scale. The advantage of such estimates lies in the invariance of the estimator when data is translated, rotated or changes in scale. Let $\boldsymbol{X}$ be an $n \times p$ dimensional data set. A location estimator $\boldsymbol{T}$ is called *affine equivariant* if and only if for all p-dimensional vectors $\boldsymbol{b}$ and all nonsingular $p \times p$ matrices $\boldsymbol{A}$ the following holds

$$\boldsymbol{T}(\boldsymbol{X}\boldsymbol{A} + \mathbf{1}\boldsymbol{b}') = \boldsymbol{T}(\boldsymbol{X})\boldsymbol{A} + \mathbf{1}\boldsymbol{b}', \tag{2.4}$$

where $\mathbf{1}$ is a column vector with n components all equal to 1.

When estimating the covariance matrix of $\boldsymbol{X}$, an estimator $\boldsymbol{C}$, which is a positive definite, symmetric $p \times p$ matrix is called *affine equivariant* if and only if

$$\boldsymbol{C}(\boldsymbol{X}\boldsymbol{A} + \mathbf{1}\boldsymbol{b}') = \boldsymbol{A}'\boldsymbol{C}(\boldsymbol{X})\boldsymbol{A} \tag{2.5}$$

holds true for all p-dimensional vectors $\boldsymbol{b}$ and all nonsingular $p \times p$ matrices $\boldsymbol{A}$.

2.1 Representation in coordinates

As stated in the introductory chapter analyzing compositional data is best done by transforming it with an *isometric log-ratio (ilr)* transformation. We focus on one particular ilr transformation with useful properties that supports our proposed algorithms. The ilr coordinates for a D-part composition $\boldsymbol{x} = (x_1, \ldots, x_D)'$ are defined as $ilr(\boldsymbol{x}) = \boldsymbol{z} = (z_1, \ldots, z_{D-1})'$, where

$$z_j = \sqrt{\frac{D-j}{D-j+1}} \ln \frac{x_j}{\sqrt[D-j]{\prod_{k=j+1}^{D} x_k}}, \quad j = 1, \ldots, D-1. \tag{2.6}$$

The *inverse isometric log-ratio (invilr)* transformation of $\boldsymbol{z} = (z_1, \ldots, z_{D-1})'$ is defined as

$$\begin{aligned}
x_1 &= \exp\left(\frac{\sqrt{D-1}}{\sqrt{D}} z_1\right),\\
x_i &= \exp\left(-\sum_{j=1}^{i-1} \frac{1}{\sqrt{(D-j+1)(D-j)}} z_j + \frac{\sqrt{D-i}}{\sqrt{D-i+1}} z_i\right), \quad i = 2, \ldots, D-1, \text{ and}\\
x_D &= \exp\left(-\sum_{j=1}^{D-1} \frac{1}{\sqrt{(D-j+1)(D-j)}} z_j\right).
\end{aligned} \tag{2.7}$$

2.2 Zeros and coordinate representation

One has to take into account that zeros might naturally occur in the data set (see Hron et al. [2015]). Zeros caused by any rounding error (this refers to so called *rounded zeros*, typically present in geochemical data) are not considered, but zeros may be the result of structural processes (*structural zeros*). Once again considering the household example, there could be teetotal households that do not have any expenditures on alcohol or tobacco. Zero values are in contradiction with the definition of compositions as data with positive entries. This is quite a natural requirement, because a multivariate observation is a composition if and only if all the relevant information is contained in the ratios between the compositional part (see Cortes [2009]). However, as a severe consequence, the log-ratio coordinates where logarithms of ratios of compositional parts are taken, cannot be applied to compositional data with zeros. There are also some alternative transformations for compositional data that avoid the problem of dealing with zero compositional parts, like the square root or the hyperspherical transformations (see Stewart and Field [2010]), resulting from considering a fixed constant sum constraint 1 of compositional parts instead of scale invariance as it is the case in the log-ratio approach. Although the transformations represent concepts of dealing with compositional data that allow for zero parts, they fail (from the perspective of the log-ratio approach) in other important features like incorporating relative scale of compositions, or their subcompositional coherence (see Cortes [2009]).

As stated in Hron et al. [2015] the advantage of our chosen log-ratio transformation as in Formula (2.6) becomes quite clear when considering a D-part composition $\boldsymbol{x}_i = (x_{i1}, \dots, x_{iD})'$, for $i = 1, \dots, n$. Suppose that $\boldsymbol{x}_i$ has $D - K(i)$ structural zeros, $2 \leq K(i) \leq D - 1$. We put the zeros on the first positions, resulting in $\tilde{\boldsymbol{x}}_i = (0, \dots, 0, x_{ij_1}, \dots, x_{ij_{K(i)}})'$. The cell x_{ij_k} corresponds to the kth non-zero position in the vector $\boldsymbol{x}_i$, for $k \in \{1, \dots, K(i)\}$. It is now straightforward to find a $D \times D$ dimensional permutation matrix $\tilde{\boldsymbol{P}}_i$ with 0/1 entries, such that $\tilde{\boldsymbol{x}}_i = \tilde{\boldsymbol{P}}_i \boldsymbol{x}_i$, for $i = 1, \dots, n$. The idea behind the re-arrangement of the parts is to construct an ilr representation of the non-zero parts. We can use Formula (2.6) for that purpose: an ilr coordinate z_{ij_k} will describe all the relative information about the part x_{ij_k} with respect to all "subsequent" parts x_{ij_l}, with $l > k$. Therefore, the corresponding ilr coordinates $z_{ij_1}, \dots, z_{ij_{K(i)-1}}$ contain all the relative information of $x_{ij_1}, \dots, x_{ij_{K(i)}}$, for $i = 1, \dots, n$. This transformation also ensures that (imputed) zeros in $\boldsymbol{x}_i$ influence the new coordinates $\boldsymbol{z}_i$ as little as possible. We will use these coordinates in the following and present two approaches to dealing with structural zeros in the data in Chapter 3.

2.3 Robust regression estimators

In this section the different robust regression methods used for detecting outliers are described. The following notation is used for the linear regression model:

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_q x_{iq} + \varepsilon_i, \quad i = 1, \dots, n$$

Often these n equations are written in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} 1 & \boldsymbol{x}_1 \\ 1 & \boldsymbol{x}_2 \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1q} \\ 1 & x_{21} & \cdots & x_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{pmatrix}, \qquad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_q \end{pmatrix}.$$

Furthermore the following assumptions are made. The ε_i are i.i.d and independent of x_i with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The residuals are denoted as $r_i(\boldsymbol{\beta}) = y_i - (\beta_0 + \beta_1 x_{i1} + \dots + \beta_q x_{iq})$.

Instead of the standard least squares method, which minimizes the sum of squared residuals

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{h} r_i(\boldsymbol{\beta})^2,$$

several robust regression methods can be formulated with positive breakdown point up to 50%. In the regression context, outliers can be defined with the help of the residuals. The i-th observation is considered an outlier, if its *standardized residual*

$$\frac{r_i(\hat{\boldsymbol{\beta}})}{\hat{\sigma}(\boldsymbol{r})} \tag{2.8}$$

exceeds/undercuts ± 2.5, where $\hat{\sigma}$ denotes the estimated standard deviation of the residuals. These boundaries are due to the model assumptions of $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ and therefore the standardized residuals are distributed according to $\mathcal{N}(0,1)$. Of course the standard deviation for robustly estimated residuals also has to be estimated in a robust way.

2.3.1 LTS-estimator

Let $|r|_{(i)}$ be the i-th ordered absolute residual, with $|r|_{(1)} \leq |r|_{(2)} \leq \ldots \leq |r|_{(n)}$. Furthermore the residual's standard deviation is estimated by

$$\sigma = \sqrt{\frac{1}{h} \sum_{i=1}^{h} |r|_{(i)}^2}.$$

The *Least Trimmed Squares* (LTS-) estimator (see Rousseeuw [1984]) is then defined as

$$\hat{\boldsymbol{\beta}}_{\text{LTS}} = \arg\min_{\boldsymbol{\beta}} \sqrt{\frac{1}{h} \sum_{i=1}^{h} |r|_{(i)}^2} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{h} |r|_{(i)}^2 .$$

The maximum breakdown point of 50% is achieved for $h = [(n+q)/2]$, but this results in an asymptotic efficiency of only 7%.

The choice of h determines both breakdown point and efficiency,

$$\left[\frac{n+q}{2}\right] \leq h \leq n,$$

with LTS being *more robust* for h towards the lower end and *more efficient* for h towards the upper end of the interval. Using $h \approx n/2$ this method has a breakdown point of approximately 50%, for larger h it depletes to about $(n-h)/n$.

2.3.2 MM-estimates

M-estimates of regression are defined as (see also Koller and Stahel [2011])

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \rho\left(\frac{r_i(\boldsymbol{\beta})}{\sigma}\right), \tag{2.9}$$

where $\rho(r)$ is assumed to be a nondecreasing function of $|r|$, with $\rho(0) = 0$ and strictly increasing for $r > 0$ where $\rho(r) < \rho(\infty)$. Maronna et al. [2006] restrict the term ρ-function to this type of functions. If ρ is bounded, it is assumed that $\rho(\infty) = 1$ and the estimate defined by (2.9) is then called *redescending* M-estimate of regression. The scale σ is required to gain scale equivariance and can either be an external scale estimate or estimated simultaneously. Differentiating (2.9) results in the estimating equation

$$\sum_{i=1}^{n} \psi\left(\frac{r_i(\hat{\boldsymbol{\beta}})}{\sigma}\right) \boldsymbol{x}_i = 0,$$

where ψ is proportional to ρ' and is usally chosen to have $\psi'(0) = 1$.

An *M-estimate of scale* of $\boldsymbol{e} = (e_1, \ldots, e_n)$ is the solution $\hat{\sigma}$ to the estimating equation

$$\frac{1}{n} \sum_{i=1}^{n} \chi\left(\frac{e_i}{\sigma}\right) = \kappa,$$

where κ is a tuning constant and $\chi(e)$ fullfills the same properties as ρ does.

S-estimates of regression are the parameter values $\hat{\boldsymbol{\beta}}_S$ that minimize the M-estimate of scale $\hat{\sigma}_S = \hat{\sigma}(\boldsymbol{r}(\boldsymbol{\beta}))$ of the associated residuals,

$$\hat{\boldsymbol{\beta}}_S = \arg\min_{\boldsymbol{\beta}} \hat{\sigma}_s(\boldsymbol{r}(\boldsymbol{\beta})).$$

The maximal breakdown point $(1 - q/n)/2$ of the S-estimate is attained at $\kappa = (1 - q/n)/2$ (see Maronna et al. [2006] for details). It is impossible for an S-estimator to achieve both a high breakdown point and a high efficiency. Following the proposal of Yohai [1987] arbitrarily high efficiency is possible by using *MM-estimates*. They are defined as the local minimum of the M-estimator for regression,

$$\hat{\boldsymbol{\beta}}_M = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \rho\left(\frac{r_i(\boldsymbol{\beta})}{\hat{\sigma}}\right),$$

obtained by using an iterative procedure started at an initial S-estimate $\hat{\boldsymbol{\beta}}_S$. The corresponding $\hat{\sigma}_S$ is used as the scaling factor in the formula above. The functions ρ and χ are usually taken from the same family. The tuning constant for ρ is determined such that the estimator reaches a desired value for the asymptotic efficiency.

2.4 Outlier detection for univariate data

When considering one-dimensional data outliers are points that are "far enough" away from the main bulk of the data. One way of locating these points is to measure location and scale of a data sample in a robust way. All observations which exceed the range of the location plus/minus multiple times the scale can be considered as potential outliers. Several methods for robustly estimating location and scale are available. We chose the ones used by Rousseeuw and Van den Bossche [2016] due to their combination of robustness and computational efficiency. For further discussion let $\boldsymbol{x} = (x_1, \dots, x_n)$ be a univariate data set.

For estimating location and scale of $\boldsymbol{x}$ the first step of an algorithm of M-estimators, as described in [Maronna et al., 2006, pp. 39-41], is used. In particular, for *robust location* estimation we start from the initial estimates, the median and a robust measure of spread,

$$m_1 = \operatorname{med}_{i=1}^n(x_i) \quad \text{and} \quad s_1 = \operatorname{med}_{i=1}^n|x_i - m_1|,$$

and then compute the location estimate

$$robLoc(\boldsymbol{x}) = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}, \tag{2.10}$$

where the weights are given by $w_i = W_{c_1}((x_i - m_1)/s_1)$. Here $W_{c_1}(\cdot)$ means Tukey's biweight function

$$W_{c_1}(t) = \left(1 - \left(\frac{t}{c_1}\right)^2\right)^2 \mathbb{1}(|t| \leq c_1),$$

where $c_1 > 0$ is a tuning constant (by default $c_1 = 3$).

For robustly estimating *scale* we assume that $\boldsymbol{x}$ has already been centered, e.g. by computing $\boldsymbol{x} - robLoc(\boldsymbol{x})$, so only the deviations from zero have to be calculated. The function

$$\rho_{c_2}(t) = \min(t^2, c_2^2)$$

is used for a constant $c_2 = 2.5$. Starting from the initial estimate $s_2 = \operatorname{med}_i(|x_i|)$ we then compute the scale estimate

$$robScale(\boldsymbol{x}) = s_2 \sqrt{\frac{1}{\delta}\frac{1}{n}\sum_{i=1}^n \frac{x_i}{s_2}}, \tag{2.11}$$

where the constant $\delta = 0.845$ ensures consistency for gaussian data. Let

$$m = robLoc(\boldsymbol{x}) \text{ and } s = robScale(\boldsymbol{x} - m),$$

as well as $\boldsymbol{z} = (\boldsymbol{x} - m)/s$ the standardization of data set $\boldsymbol{x}$. x_i is then considered an outlier if

$$|z_i| > \sqrt{\chi^2_{1,p}}.$$

The probability p is often chosen as 99%, so that under ideal circumstances only 1% of the entries get flagged.

2.5 Outlier detection for multivariate data

In the multivariate case potential outliers will primarily be data points, that are not in correspondence with the structure of the main bulk of the data. We use two methods to detect outliers in multivariate data, the first one being the very prominent measure of squared Mahalanobis distances MD_i^2. Let $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ be a data matrix with observations $\boldsymbol{x}_i \in \mathbb{R}^p, i = 1, \ldots, n$.

2.5.1 Robust multivariate distances

The *robust squared Mahalanobis distance* for the i-th observation is then defined by

$$RD_i^2 = (\boldsymbol{x}_i - \boldsymbol{T})'\boldsymbol{C}^{-1}(\boldsymbol{x}_i - \boldsymbol{T}), \tag{2.12}$$

where $\boldsymbol{T}$ is a robust measure of location of the data set $\boldsymbol{X}$ and $\boldsymbol{C}$ is a robust estimate of the covariance matrix. In case of $\boldsymbol{X}$ following a multivariate normal distribution, the squared classic Mahalanobis distance (based upon the sample mean and covariance matrix) follows a χ^2_p distribution (e.g. Johnson and Wichern [1998]). A common rule is to declare observations as potential outliers, if they exceed the 97.5% quantile of the chi-squared distribution $\chi^2_{p;0.975}$. In the literature one can find many different robust estimates for location $\boldsymbol{T}$ and covariance $\boldsymbol{C}$, which not only differ in breakdown points, but also in statistical efficiency. More developed robust estimates may be tuned to achieve high breakdown point combined with high efficiency.

Here we consider the *minimum covariance determinant* (MCD) estimator, first introduced by Rousseeuw [1985]. This particular estimator features affine equivariance for location and scale and can achieve the maximal breakdown point of 50%. Given a data set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ with $\boldsymbol{x}_i \in \mathbb{R}^p, i = 1, \ldots, n$, the minimum covariance determinant estimator is defined by the subset of h data points $\{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_h}\}$ with the smallest determinant of the sample covariance matrix among all subsets of size h. The MCD estimate for location $\boldsymbol{T}_{MCD}$ and covariance $\boldsymbol{C}_{MCD}$ is then defined by

$$\boldsymbol{T}_{MCD} = \frac{1}{h}\sum_{i=1}^{h} \boldsymbol{x}_{i_j}, \tag{2.13}$$

$$\boldsymbol{C}_{MCD} = c_{ccf}c_{sscf}\frac{1}{h-1}\sum_{j=1}^{h}(\boldsymbol{x}_{i_j} - \boldsymbol{T}_{MCD})(\boldsymbol{x}_{i_j} - \boldsymbol{T}_{MCD})'. \tag{2.14}$$

The coefficients c_{ccf} and c_{sscf} are correction factors. The consistency correction factor c_{ccf} is chosen such that $\boldsymbol{C}_{MCD}$ is consistent at the multivariate normal model, however this factor is only reliable for subset size h close to n. Otherwise this factor produces overestimation. The small sample correction factor c_{sscf} is chosen such that the estimate is unbiased at samples with small number of observation. Further information on the correction factors may be found in Rocke [1996], Croux and Haesbroeck [1999], Pison et al. [2002]. Parameter h is used to regulate the breakdown point of the MCD estimator. For h equal to the sample size n, the MCD estimate reduces to the classical estimate for location and scale, which inherit a breakdown point of 0%. For h equal to $\lfloor (n+p+1)/2 \rfloor$, the maximum possible breakdown point of 50% can be achieved. Often the amount of contamination in data sets is not very high, therefore $h = 0.75n$ is used in practice to get a sufficient breakdown point and as well as still good statistical properties.

In general the MCD estimate is not very efficient, especially if h is chosen in order to achieve the maximum breakdown point. A way to overcome the low efficiency of the MCD estimator is by introducing an adaptation of the estimate including sample weights. The weights $w_i, i = 1, \ldots, n$, are introduced in such a way that

$$w_i = \begin{cases} 1 & \text{if } (\boldsymbol{x}_i - \boldsymbol{T}_{MCD})' \boldsymbol{C}_{MCD}^{-1} (\boldsymbol{x}_i - \boldsymbol{T}_{MCD}) \leq \chi^2_{p,0.975} \\ 0 & \text{otherwise} \end{cases},$$

where $\boldsymbol{T}_{MCD}$ and $\boldsymbol{C}_{MCD}$ are the original MCD estimates for location and scale. The reweighted MCD estimates $\boldsymbol{T}_{MCDr}$ and $\boldsymbol{C}_{MCDr}$ are given by

$$\boldsymbol{T}_{MCDr} = \frac{1}{v} \sum_{j=1}^{n} \boldsymbol{x}_{i_j},$$

$$\boldsymbol{C}_{MCDr} = c_{r,ccf} c_{r,sscf} \frac{1}{v-1} \sum_{i=1}^{n} w_i (\boldsymbol{x}_{i_j} - \boldsymbol{T}_{MCDr})(\boldsymbol{x}_{i_j} - \boldsymbol{T}_{MCDr})',$$

where v equals the sum of all weights $v = \sum_{i=1}^{n} w_i$. As with the original MCD estimate, the factors $c_{r,ccf}$ and $c_{r,sscf}$ are correction factors to achieve consistency and unbiasedness for small samples. The reweighted and the initial MCD estimator feature identical breakdown point, but the efficiency of the former is higher. The big advantage of the MCD estimate and the reason for its popularity among robust estimators is the fast algorithm for its computation.

2.5.2 *sign*-method

As a second method to detect multivariate outliers we consider the *sign*-method as proposed by Filzmoser et al. [2008]. This procedure utilizes a robust *principal component analyis* (PCA) implementation by Locantore et al. [1999].

Let us first recapitulate the basic concept of PCA (e.g as described in Hastie et al. [2001]). Let $\boldsymbol{X}$ be the centered $n \times p$ dimensional data matrix. The *singular value* *decomposition* (SVD) of $\boldsymbol{X}$ is then defined as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}', \tag{2.15}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $n \times p$ and $p \times p$ dimensional orthogonal matrices, with the columns of $\boldsymbol{U}$ spanning the column space of $\boldsymbol{X}$ and the columns of $\boldsymbol{V}$ spanning the row space. $\boldsymbol{D}$ is a $p \times p$ dimensional diagonal matrix, with diagonal entries $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$ called the *singular values* of $\boldsymbol{X}$. If one or more values $d_j = 0$, $\boldsymbol{X}$ is singular.

The SVD of the centered matrix $\boldsymbol{X}$ is another way of expressing the *principal components* of the variables in $\boldsymbol{X}$. The sample covariance matrix is given by $\boldsymbol{C} = \boldsymbol{X}'\boldsymbol{X}/n$, and from Formula (2.15) we have

$$\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}', \tag{2.16}$$

which is the *eigen decomposition* of $\boldsymbol{X}'\boldsymbol{X}$ (and of $\boldsymbol{C}$, up to a factor n). The *eigen vectors* (columns of $\boldsymbol{V}$) are also called the principal components directions of $\boldsymbol{X}$. The first principal component direction v_1 has the property, that $\boldsymbol{z}_1 = \boldsymbol{X}v_1$ has the largest sample variance amongst all normalized linear combinations of the columns of $\boldsymbol{X}$. One can easily see that this sample variance is

$$\mathrm{Var}[\boldsymbol{z}_1] = \mathrm{Var}[\boldsymbol{X}v_1] = \frac{d_1^2}{n}.$$

The derived variable $\boldsymbol{z}_1$ is called the *first principal component* of $\boldsymbol{X}$. Subsequent principal components $\boldsymbol{z}_j$ have maximum variance d_j^2/n, subject to being orthogonal to the earlier ones. Conversely the last principal component has minimum variance. Hence the small singular values d_j correspond to directions in the column space of $\boldsymbol{X}$ having small variance.

Continuing, Filzmoser et al. [2008] propose the following algorithm for the *sign*-method:

Step 1: Robustly sphere the data by subtracting the median and dividing by the *median absolute deviation* (MAD), the latter being defined for a sample $\{x_1, \ldots, x_n\} \subset \mathbb{R}$ as

$$\mathrm{MAD}(x_1, \ldots, x_n) = 1.4826 \cdot \mathrm{med}_j |x_j - \mathrm{med}_i x_i|, \tag{2.17}$$

in each dimension. Project the data onto the unit sphere by standardizing each data point by its norm, $x^*_{ij} = \frac{x_{ij}}{\|\boldsymbol{x}_i\|}$.

Step 2: Due to Step 1, the effect of outlying observations is greatly minimized and simply computing the classical covariance matrix yields a robust covariance matrix $\boldsymbol{C}$. It follows that performing standard PCA on $\boldsymbol{C}$ yields a robust principal components decomposition. Therefore we calculate the sample covariance matrix of the sphered data followed by the (classical) principal components $\boldsymbol{z}_j \in \mathbb{R}^n, j = 1, \ldots, p$.

Step 3: Determine and retain only those p^* components that contribute to at least

99% of the variance. Robustly sphere the data in this modified PC space according to

$$z_{ij}^{*} = \frac{z_{ij} - \text{med}_i z_{ij}}{\text{MAD}_i z_{ij}}, \quad j = 1, \ldots, p^{*}.$$

Step 4: Calculate robust distances from these data according to

$$RD_i = \sqrt{\sum_{j=1}^{p^{*}} z_{ij}^{*2}}, \quad i = 1, \ldots, n,$$

which is computationally fast in PC space.

Step 5: Transform these distances according to

$$d_i = RD_i \cdot \frac{\sqrt{\chi^2_{p^{*},0.5}}}{\text{med}_i(RD_i)}$$

to better approximate a $\chi^2_{p^{*}}$ distribution and classify as outliers all points with transformed robust distance greater than the $\chi^2_{p^{*}}$ 0.975 quantile.

2.6 Regression methods for outlier detection

Summarizing the previous sections we have become acquainted with two fundamental methods to detect outliers in multivariate data, being standardized residuals in a regression setting and robust distances. In regression diagnostics these two concepts are often combined to distinguish between four different types of data points:

- *regular observations*: Observations that fit the regression line well in both dimension (y- & X-dimension).
- *vertical outliers*: Observations that have outlying values for the corresponding error term (the y-dimension) but are not outlying in the design space (the X-dimension).
- *good leverage points*: Observations that are outlying in the design space, but are located close to the regression line.
- *bad leverage points*: Observations located far away from the regression line.

In general, good leverage points yield an advantage due to them lying within the direction of the regression hyperplane. Thus they contribute to an even more precise estimation of the regression parameters. However bad leverage points may have strong influence on parameter estimation, because they might even tilt the regression hyperplane.

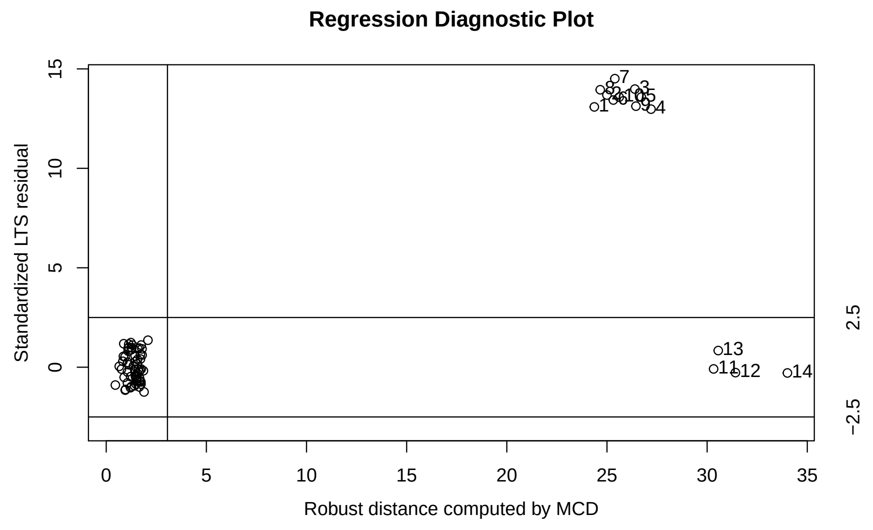

Figure 2-1: Regression diagnostic plot of the Hawkins-Bradu-Krass data set.

As stated in Formula (2.8) outliers in regression are identified by large standardized residuals (outside of the bounds ±2.5). Leverage points are outliers in the X-space and therefore have large robust distances. The *regression diagnostic plot* compares standardized robust residuals and robust distances. To illustrate this we consider an artificial data set generated by Hawkins et al. [1984]. The data set consists of 75 observations in four dimensions (one response and three explanatory variables). It provides a good example of the masking effect. The first 14 observations are outliers, created in two groups: 1–10 and 11–14. Only observations 12, 13 and 14 appear as outliers when using classical methods, but can be easily unmasked using robust distances. Figure 2-1 shows the regression diagnostic plot resulting from plotting the robust standardized residuals resulting from an LTS-regression versus the robust distances obtained from using robust MCD estimates. Regular observations as well as good leverage points can be found within the ±2.5 bound of the standardized residuals. As described in Section 2.5.1 the robust distance of an observation determines whether it is a good or bad leverage point (based on $\sqrt{\chi^2_{q;0.975}}$). The diagnostic plot shows that the outlying observations 11-14 are actually good leverage points and thus help to stabilize the regression hyperplane. Observations 1-10 are successfully identified as outliers.

We try to utilize this method of outlier identification in Chapter 3 for detecting deviating cells within the data set. However the compositional structure of the data and the thus needed transformations before the actual analysis require proficient adaptation of this concept.

Chapter 3

Cellwise outlier detection in compositional data

As outlined in the introductory chapter of this thesis we are interested in detecting cellwise and rowwise outliers on the simplex, the original sample space induced by compositional data. For this purpose we use

– the isometric log-ratio transformation introduced in Section 2.1 and

– the pairwise log-ratio matrix introduced in Section 3.3

to analyze the ratios between the parts of an observation and detect deviating behavior. When using log-ratios one has to take into account that zeros might naturally occur in the data set. We discussed this issue in Section 2.2 and utilize

– robust imputation methods as depicted in Appendix A and

– splitting the data set into subsets according to its zero patterns

to deal with this issue. Combining these methods we propose four different approaches for detecting outliers within the data:

– imputation approach (see Section 3.1)

– subset approach (see Section 3.2)

– pairwise log-ratio approach (see Section 3.3)

– detect deviating cells (algorithm proposed in Rousseeuw and Van den Bossche [2016]) on pairwise log-ratios (see Section 3.4)

As stated in Section 2.6 we combine the concepts of

– robust regression and

– robust distance calculation

to analyze our data. One could argue that after dealing with structural zeros and representing the variables in the new coordinates, we could simply use robust distances to detect deviating observations within the ratios. However robust regression allows us to detect outliers in a regression setting, analyzing standardized residuals instead of distances. Combining the information provided by the residuals with the one obtained from the robust distances even allows for cellwise outlier detection. More importantly the regression approach enables an easy integration of external, non-compositional variables (e.g. demographic information), possibly gaining valuable information.

3.1 Imputation approach

The main idea of the algorithm is to firstly deal with missing values and structural zeros by imputing them using robust imputation methods (such as described in Appendix A and Templ et al. [2015]). The major advantage of these imputation methods is due to the fact, that observations containing zero parts are imputed in a way such that no outliers are produced. We can therefore use the full information without worrying about zeros or additional outliers. Due to the compositional structure of the data the compositional variables (excluding additional grouping- and regression-variables) are expressed into the new coordinates using an isometric log-ratio transformation (see Equation (2.6)). Afterwards robust, cellwise regression, taking the transformed as well as grouping- and additional variables into account, can be used to calculate standardized residuals. Although cellwise regression is applied on the data set, the resulting standardized residuals as in Formula (2.8) are not suitable for a cellwise outlier detection yet. Looking at the formula for the residuals in a standard regression

$$r_i(\boldsymbol{\beta}) = y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_q x_{iq}),$$

large absolute residuals can either derive from outlying values in the response variable y_i or from the regressors $x_{ij},\ j = 1, \ldots, q$. Naturally this fact applies to the standardized (robust) residuals as well, making it ambiguous to tell if the detected outlier indeed stems from the cell y_i only. Therefore, for each regression, also robust Mahalanobis distances within the space of the regressor variables are computed. If the robust distance of the regressors of this particular observation is in line with the majority of the data, the large residual may only stem from y_i alone. By combining the results of residuals and distances one may detect cellwise outlier in the original, untransformed sample space. However, due to the nature of the isometric log-ratio transformation, where the new coordinates always inherit parts of the original variables, this proves to be quite difficult.

3.1.1 Detailed description of the algorithm

Let $(\boldsymbol{X}, \boldsymbol{Y})$ be our initial $n \times (p+D)$ dimensional data set, where $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ denotes the compositional variables, $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_D)$ the non-compositional variables of the set.

Step 1: imputation. In case of compositional data, structural zeros are regarded as missing information. In order to work with imputation methods (e.g. as presented in Appendix A), all zeros in the compositional data $\boldsymbol{X}$ are set to missing beforehand. Afterwards $\boldsymbol{X}$ is imputed using robust imputation methods resulting in $\boldsymbol{X}^{imp}$.

Step 2: iteration over columns. In order to detect univariate outliers within the observations of $\boldsymbol{X}$, which lead to outlying ratios, the algorithm sequentially applies transformations and outlier detection on sorted data sets. Therefore we iterate over the columns $\boldsymbol{x}_j$ for all $j \in \{1, \ldots, p\}$, calculating:

2.1: sorting. Sort $\boldsymbol{X}^{imp}$ so that $\boldsymbol{x}_j$ is the first column of $\boldsymbol{X}^{imp}$.

2.2: representation in coordinates. Apply the isometric log-ratio transformation from Equation (2.6) on the sorted $\boldsymbol{X}^{imp}$ to obtain $\boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{p-1})$.

2.3: robust regression. Apply robust regression

$$\boldsymbol{z}_1 = (\boldsymbol{Z}_{(-1)}, \boldsymbol{Y})\boldsymbol{\beta}^{(j)} + \boldsymbol{\varepsilon}^{(j)},$$

where $\boldsymbol{Z}_{(-1)}$ depicts the obtained data set $\boldsymbol{Z}$ excluding the first column, and $\boldsymbol{\beta}^{(j)}$, $\boldsymbol{\varepsilon}^{(j)}$ the regression coefficients and residuals of this particular regression.

2.4: calculate standardized residuals. Calculate standardized residuals as in Equation (2.8)

$$\boldsymbol{r}^{(j)} = \frac{\hat{\boldsymbol{\varepsilon}}^{(j)}}{\hat{\sigma}(\hat{\boldsymbol{\varepsilon}}^{(j)})},$$

where $\hat{\sigma}$ denotes the estimated standard deviation of the residuals.

2.5: calculate robust distances in the space of the regressor variables. Calculate robust squared Mahalanobis distances for each observation/row $(\boldsymbol{z}_{i\cdot}, \boldsymbol{y}_{i\cdot})$, $i = 1, \ldots, n$ of the regressor matrix $(\boldsymbol{Z}_{(-1)}, \boldsymbol{Y})$ using Formula (2.12)

$$d_i^{(j)} = ((\boldsymbol{z}_{i\cdot}, \boldsymbol{y}_{i\cdot}) - \boldsymbol{T}^{(j)})'(\boldsymbol{C}^{(j)})^{-1}((\boldsymbol{z}_{i\cdot}, \boldsymbol{y}_{i\cdot}) - \boldsymbol{T}^{(j)}),$$

where $\boldsymbol{T}^{(j)}$ is a robust measure of location of these particular regressor variables $(\boldsymbol{Z}_{(-1)}, \boldsymbol{Y})$ and $\boldsymbol{C}^{(j)}$ is a robust estimate of the covariance matrix. In our case the MCD estimates for location and scale as described in Section 2.5.1 are used.

Iteration over all columns $\boldsymbol{x}_j$, $j = 1, \ldots, p$ of $\boldsymbol{X}^{imp}$ results in standardized residuals $\boldsymbol{r}^{(j)} \in \mathbb{R}^n$ for each column of $\boldsymbol{X}^{imp}$. Furthermore, we obtain robust distances $\boldsymbol{d}^{(j)} \in \mathbb{R}^n$ of the regressor variables of the j-th regression. One has to note, that the residuals as well as the distances are based on the corresponding ratios of $\boldsymbol{X}$.

Step 3: outlier detection. Flag each cell of $\boldsymbol{X}$ as outlier according to

$$f_i^{(j)} := \mathbb{1}(|r_i^{(j)}| > c),$$

for all observations $i \in \{1, \dots, n\}$ and compositional variables $j \in \{1, \dots, p\}$. Often $c = 2.5$ is chosen. Note that due to the definition of the isometric log-ratio transformation in Formula (2.6), the first coordinate $\boldsymbol{z}_1$ is not only influenced by $\boldsymbol{x}_j$ (in the j-th run of Step 2), but is also influenced by all other variables $\boldsymbol{x}_k,\ k \in \{1, \dots, p\},\ k \neq j$. Therefore large absolute standardized residuals in $\boldsymbol{r}^{(j)}$ and hence an outlier flag in $\boldsymbol{f}^{(j)}$ may be the result of outlying values in $\boldsymbol{x}_k,\ k \neq j$.

As a consequence we also have to consider the robust distances $\boldsymbol{d}^{(j)}$ of the regressor variables of the j-th regression. Here we use robust mahalanobis distances as described in Section 2.5.1. These distances are only influenced by $(\boldsymbol{Z}_{(-1)}, \boldsymbol{Y})$, which do not stem from $\boldsymbol{x}_j$ at all. In general, as stated in Section 2.5.1, distances of observations are considered outlying if

$$g_i^{(j)} := \mathbb{1}(d_i^{(j)} > \chi^2_{p-2+D;0.975})$$

for all observations $i \in \{1, \dots, n\}$ and compositional variables $j \in \{1, \dots, p\}$.

Step 4: interpretation. The following points are to consider when interpreting the above results:

4.1: cellwise outlier. Following our findings, a cellwise outlier in $\boldsymbol{X}$ is a cell x_{ij}, whose outlier flag $f_i^{(j)} = 1$ and whose distance flag $g_i^{(j)} = 0$. This constellation indicates, that in the j-th run of Step 2, x_{ij} solely contributed to a deviating ratio z_{i1}, resulting in a large absolute standardized residual $r_i^{(j)}$ in the j-th regression. This residual is not influenced by any other deviating x_{ij}, due to the robust Mahalanobis distances $d_i^{(j)}$ not indicating any outlyingnes in the regressor space, which is solely comprised of variables independent of $\boldsymbol{x}_j$.

4.2: multivariate/rowwise outlier Step 4.1 immediately gives the definition of a multivariate outlier for this algorithm. If outlier flags $f_i^{(j)}$ indicate large absolute standardized residuals for observation i, and the corresponding robust distances $g_i^{(j)}$ also show deviating results in the regressor space, several x_{ij} are responsible for a deviating behaviour of the ratios of this particular observation. In this case outlier flags may be summed up

$$\sum_{j=1}^{p} f_i^{(j)} \quad \text{for the } i\text{-th observation},$$

indicating how often this observation was identified as an outlier. Based on this sum the whole row could be flagged as an outlier. We also flag the whole observation as an outlier, if for each regression $j = 1, \ldots, p$ the robust distance flag $g_i^{(j)}$ indicates a multivariate outlier in the regressor space, regardless of the residual flag. This predominantly indicates a multivariate outlier within this observation. Therefore we flag observation i as a multivariate outlier if

$$\sum_{j=1}^{p} g_i^{(j)} = p. \tag{3.1}$$

3.1.2 Advantages and limitations of the algorithm

Recapitulating the above, with this approach zeros are imputed first. Imputation is done in a way, that no additional outliers are generated when filling missing information. The big advantage of the imputation approach is to work with n observations, while in the subset approach (see Section 3.2) the zero-patterns are taken into account, naturally working on smaller subsets of the data set then. The algorithm further gains advantage over the pairwise approaches (see Sections 3.3 & 3.4) by using additional variables for the regression, possibly providing additional information concerning the ratios. With this approach, cellwise outlier detection proves to be quite difficult. Due to the nature of the isometric log-ratio transformation, observations represented in the new coordinates are almost always influenced by possible multivariate outliers in the original sample space. This makes it quite difficult to deduce which cell/cells of the observation in the untransformed space leads/lead to outlying ratios. However as described in Step 4.1, proficient interpretation of the regression- and robust distance results may track down an individual cellwise outlier in the original sample space, leading to deviating ratios. Concluding, it is important to note (as described in Step 4.2) that due to the multiple regressions resulting in multiple outlier flags per observations, the algorithm gives a good indication of observations whose ratios deviate from the main bulk of the data.

3.2 Subset approach

Contrary to the imputation approach discussed in the previous section, the subset approach handles missing values by dividing the data set into unique subsets (patterns) according to the zero-pattern occurring in each row. Therefore imputation is not needed and the process of representing the variables in the new coordinates using the isometric log-ratio transformation and then performing a cellwise regression can be executed on each subset. Using the methods described in the imputation approach, robust standardized residuals and robust distances are calculated for each subset of the data set and results may again be combined to try to detect cellwise outliers in the original, untransformed sample space. Due to the subset approach directly resembling the imputation approach, this again poses to be a difficult task.

3.2.1 Detailed description of the algorithm

Let $(\boldsymbol{X}, \boldsymbol{Y})$ again be our initial $n \times (p+D)$ dimensional data set, where $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_p)$ denotes the compositional variables, $\boldsymbol{Y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_D)$ the non-compositional variables of the set.

Step 1: split into subsets according to zero pattern. We split the data set $(\boldsymbol{X}, \boldsymbol{Y})$ into unique subsets $(\boldsymbol{X}_k, \boldsymbol{Y}_k)$, $k = 1, \dots, l$ with n_k observations and $(p+D)$ columns, where $\sum_{k=1}^{l} n_k = n$, according to the l zero patterns in $\boldsymbol{X}$.

Step 2: iteration over subsets. Basically we now apply Steps 2-4 of the algorithm described in Section 3.1.1 on the subsets $(\boldsymbol{X}_k, \boldsymbol{Y}_k)$ instead of the whole, imputed data frame. Therefore we iterate over the subsets for all $k \in \{1, \dots, l\}$, calculating:

2.1 delete zero columns. Delete all p_k zero columns in $\boldsymbol{X}_k$ resulting in a data set $\widetilde{\boldsymbol{X}}_k$ with $\tilde{p}_k := p - p_k$ columns.

2.2 validate subset. Subsets now have to be re-evaluated concerning their sample size. For meaningful regression the subset $(\widetilde{\boldsymbol{X}}_k, \boldsymbol{Y}_k)$ must contain more than $2 \cdot (\tilde{p}_k + D) + 1$ observations. If this is not the case for the present subset, Step 2.3 is skipped entirely and possible outliers in this subset can not be detected. Outlier- and distance-flags $f_i^{(j)}, g_i^{(j)}$ are set to 0 for all rows in the subset.

2.3 iteration over columns of subset. We now iterate over the columns $\tilde{\boldsymbol{x}}_j$ of subset $\widetilde{\boldsymbol{X}}_k$ for all $j \in \{1, \dots, \tilde{p}_k\}$ and perform the exact same Steps 2.1-2.5 of the imputation algorithm (see Section 3.1.1), whereas in this case we use

- $\widetilde{\boldsymbol{X}}_k$ instead of $\boldsymbol{X}^{imp}$ and
- $\boldsymbol{Y}_k$ instead of $\boldsymbol{Y}$.

Iteration over all columns $\tilde{\boldsymbol{x}}_k$, $j = 1, \dots, p_k$ of $\widetilde{\boldsymbol{X}}_k$ results in standardized residuals $\boldsymbol{r}^{(j)} \in \mathbb{R}^{n_k}$ for each column of $\widetilde{\boldsymbol{X}}_k$ or for each non-zero column of $\boldsymbol{X}_k$ respectively. Furthermore, we obtain robust distances (here we use the sign-method introduced in Section 2.5.2 for calculation) $\boldsymbol{d}^{(j)} \in \mathbb{R}^{n_k}$ of the regressor variables of the j-th regression.

Step 3: combining results. After iterating over the unique subsets $(\boldsymbol{X}_k, \boldsymbol{Y}_k)$ of $(\boldsymbol{X}, \boldsymbol{Y})$ we obtained standardized residuals $r_i^{(j)}$, $(i,j) \in \{(i,j) \mid i = 1, \dots, n \wedge j = 1, \dots, p \ \wedge \ x_{ij} \neq 0\}$ for every non-zero cell of $\boldsymbol{X}$, as well as the corresponding robust distances $d_i^{(j)}$.

Step 4: outlier detection. Outlier detection proceeds as in Step 3 of the imputation algorithm and results in outlier- and distance-flags $f_i^{(j)}, g_i^{(j)}$ for all observations $i = \{1, \dots, n\}$ and compositional variables $j = \{1, \dots p\}$ where $x_{ij} \neq 0$.

Step 5: interpretation. The above results can basically be interpreted as in Step 4 of the imputation approach.

5.1: cellwise outlier. A cellwise outlier in $\boldsymbol{X}$ is a cell x_{ij}, whose outlier flag $f_i^{(j)} = 1$ and whose distance flag $g_i^{(j)} = 0$. This constellation indicates, that in the j-th run of Step 2, x_{ij} solely contributed to a deviating ratio z_{i1}, resulting in a large absolute standardized residual $r_i^{(j)}$ in the j-th regression. This residual is not influenced by any other deviating x_{ij}, due to the robust Mahalanobis distances $d_i^{(j)}$ not indicating any outlyingness in the regressor space, which is solely comprised of variables independent of $\boldsymbol{x}_j$.

5.2: multivariate/rowwise outlier Step 5.1 again immediately gives the definition of a multivariate outlier for this algorithm. If outlier flags $f_i^{(j)}$ indicate large absolute standardized residuals for observation i, and the corresponding robust distances $g_i^{(j)}$ also show deviating results in the regressor space, several x_{ij} are responsible for a deviating behaviour of the ratios of this particular observation. In this case outlier flags may be summed up

$$\sum_{j=1}^{p} f_i^{(j)} \quad \text{for the } i\text{-th observation,}$$

indicating how often this observation was identified as an outlier. Based on this sum the whole row could be flagged as an outlier. Resembling the interpretation from the imputation approach, if for an observation i each calculation of robust distances leads to a positive outlier flag $g_i^{(j)}$, we conclude that this as an indication for the presence of a multivariate outlier. This is the case if

$$\sum_{j=1}^{p} g_i^{(j)} = \tilde{p}_k \quad \text{for the } i\text{-th observation belonging to subset } \tilde{\boldsymbol{X}}_k. \tag{3.2}$$

In theory, outlier detection is feasible for all observations when using the subset approach. However, as stated in Step 2.2 of the algorithm, some subsets may turn out to be too small for regression- and distance analysis, making outlier detection within these observations impossible.

3.2.2 Advantages and limitations of the algorithm

Instead of imputing zeros, this approach divides the underlying data set into subsets (patterns) according to the zero-pattern occurring in each row. The big advantage of the subset approach is that one does not have to be concerned about any imputations, as done in the previous approach. However subsets may turn out to be too small for our regression- and distance-based outlier detection methods and potential observations deviating from the main bulk of the data, can thus not be detected within those subsets. Furthermore our regression- and distance estimators are naturally working on smaller data sets, making them not as potent as if they were estimated on the full data set. However, as in the imputation approach regression analysis also takes additional variables into account, thus possibly providing additional information which pairwise approaches are missing out on. Because the subset approach basically performs the imputation approach on smaller data sets, cellwise outlier detection still proves to be quite difficult. The problems of the isometric log-ratio transformation being influenced by multivariate outliers and interpreting the results in a way to trace back cellwise outliers in the original sample space still remain. However also the subset approach can be of good use, when flagging whole observations as outliers. Multiple outlier flags per observation again may be summed up per observation, indicating how often this particular row was identified as an outlier.

3.3 Pairwise log-ratio approach

The algorithms presented so far have difficulties to detect cellwise outliers. These difficulties are related to representing the compositional variables of our data sets in the new coordinates using the isometric log-ratio transformation. Thus we present an algorithm that tries to detect cellwise outliers, which lead to deviating ratios by observing pairwise log-ratios between the variables.

In this as well as in the following Section 3.4 we are therefore making extensive use of the *pairwise log-ratio matrix* $\boldsymbol{Z}$ between the variables of a matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$, defined as

$$\boldsymbol{Z} = \left(\log \frac{\boldsymbol{x}_1}{\boldsymbol{x}_2}, \cdots, \log \frac{\boldsymbol{x}_1}{\boldsymbol{x}_p}, \log \frac{\boldsymbol{x}_2}{\boldsymbol{x}_3}, \cdots, \log \frac{\boldsymbol{x}_2}{\boldsymbol{x}_p}, \cdots, \log \frac{x_{p-1}}{x_p} \right). \tag{3.3}$$

Note that the matrix does not include ratios containing (in the regression context) duplicate ($\log \frac{x_i}{x_j} = -\log \frac{x_j}{x_i}$) or unnecessary ($\log \frac{x_i}{x_j},\ i = j$) information.

The main idea of the algorithm stems from the presumption, that large differences between the parts of an observation lead to deviating ratios. To illustrate this, let us consider a toy data set (see Table 3-1) consisting entirely of ones, with a single observation having two contaminated cells. If we were now to compute the pairwise

row.num	x1	x2	x3	x4
1	1	1	1	1
2	1	1	1	1
⋮	⋮	⋮	⋮	⋮
9	1	1	1	1
10	1	1	5	5

Table 3-1: Toy data set with cellwise contamination in observation 10.

log-ratios between the parts of each observation, observations 1-9 would all have ratios 0 between their variables $x1, \ldots, x4$. Calculating the log-ratios between the variables of observation 10 however would lead to deviating log-ratios e.g. for variables $x1$ and $x4$, $\log(x_{10,1}/x_{10,4}) \approx 1.61$. The example shows that flagging cells $x_{10,4}$ and $x_{10,5}$ of observation 10 would indicate cellwise outliers in the original sample space, which lead to deviating ratios. With this in mind we will speak of “good" variables ($x1$, $x2$) and “bad” variables ($x3$, $x4$) on an observation basis and presume that bad variables lead to deviating ratios when compared to good variables.

Taking this into account, the algorithm tries to distinguish between good and bad variables for each observation. To do this we impute compositional variables to deal with structural zeros and missing information. Then column-wise univariate outlier detection is performed on the data set. On an observation level we detect “bad” variables, if the variable was detected as a univariate outlier within the column. We treat all other variables of an observation as “good” variables. We then compare the pairwise log-ratios between bad and good variables against all ratios between the good variables. If the ratio containing a bad variable deviates (based on regression and robust distances), we conclude that the bad variable is responsible for the deviating ratio. Regression and distance methods are performed on subsets containing observations, which feature the same univariate outlier pattern, and results are combined to detect cellwise, bivariate and multivariate outliers within observations.

3.3.1 Detailed description of the algorithm

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_p)$ be the $n \times p$ dimensional compositional data set. The algorithm uses the initial univariate outlier detection as described in Rousseeuw and Van den Bossche [2016].

Step 1: imputation. We deal with structural zeros within the compositional data by setting them to missing beforehand. Afterwards $\boldsymbol{X}$ is imputed using robust imputation methods resulting in $\boldsymbol{X}^{imp}$.

Step 2: compute intial ratios. In this section’s introductory example the univariate outlier detection and the following split between “good” and “bad” variables was based on the absolute amounts of the particular cells. To account for the compositional structure of the data, also these intial steps should have ratios in mind. We therefore calculate $\boldsymbol{X}^{init}$ as

$$\boldsymbol{x}_j = \frac{\boldsymbol{x}_j}{\sum_{i \neq j} \boldsymbol{x}_i} \quad \text{for each column } j = 1, \dots, p.$$

Step 3: standardization. For each column $\boldsymbol{x}_j,\ j = 1, \dots, p$ of $\boldsymbol{X}^{init}$ we estimate

$$m_j = robLoc(\boldsymbol{x}_j) \text{ and } s_j = robScale(\boldsymbol{x}_j - m_j),$$

where *robLoc* is a robust estimator of location and *robScale* is a robust estimator of scale, which assumes its argument has already been centered (we consider the robust estimators of location (2.10) and scale (2.11) as described in Section 2.4. We then standardize $\boldsymbol{X}^{imp}$ to $\boldsymbol{Z}$ by

$$\boldsymbol{z}_j = \frac{\boldsymbol{x}_j - m_j}{s_j}.$$

Step 4: apply univariate outlier detection to all variables. After the column-wise standardization in Step 2 we compute a new matrix $\boldsymbol{U}$ with entries

$$u_{ij} = \mathbb{1}\left(|z_{ij}| > \sqrt{\chi^2_{1,p}}\right).$$

The probability p is often chosen as 99% so that under ideal circumstances only 1% of the entries get flagged. We obtained a 0-1-matrix $\boldsymbol{U}$, indicating column-wise univariate outlier.

Step 5: detect unique outlier patterns. The rows of matrix $\boldsymbol{U}$ give information about the univariate outliers occuring in each observation of $\boldsymbol{X}$. We now split $\boldsymbol{U}$ into unique subsets $\boldsymbol{U}_k$, $k = 1, \dots, l$ with n_k observations and p columns, where $\sum_{k=1}^{l} n_k = n$, according to the l outlier patterns in $\boldsymbol{U}$. Each observation of $\boldsymbol{X}$ is now uniquely assigned to a subset $\boldsymbol{U}_k$. Let I_k be the index set of observations belonging to group $\boldsymbol{U}_k$, with $\#I_k = n_k$.

Step 6: iteration over subsets. For every unique outlier pattern $\boldsymbol{U}_k$, $k = 1, \dots, l$ we calculate:

6.1: distinguish between "good" and "bad" variables. Our presumption in this algorithm is in general, with a few exceptions discussed within this section of the thesis, that univariate outliers within an observation are predominantly responsible for outlying ratios. We therefore distinguish between "good" and "bad" variables/columns within the subset $\boldsymbol{U}_k$ (due to all rows of $\boldsymbol{U}_k$ showing the same 0-1-/outlier pattern, it is sufficient to consider only the first row of $\boldsymbol{U}_k$ in the following definitions):

$$J_{good} := \{j \mid u_{1j} = 0\}$$
$$J_{bad} := \{j \mid u_{1j} = 1\}$$

Let $p_{good} := \#J_{good}$ and $p_{bad} := \#J_{bad}$. It follows that $p_{good} + p_{bad} = p$ and we have divided the subset into two "types" of variables.

6.2: ratio analysis. In order to get a good understanding of the ratios between variables and their interconnection we now apply a number of procedures on the pairwise log-ratios of the variables of $\boldsymbol{X}^{imp}$.

6.2.1: regression analysis - bad/good ratios. First we calculate the pairwise log-ratio matrix $\boldsymbol{Z}^{(k)}_{good}$ as in Formula (3.3) between all good variables $\boldsymbol{x}_j$, $j \in J_{good}$ and call it the matrix of "good ratios". We then test if the ratios between good and bad variables deviate from the good ratios. This is done by regressing all pairs of log-ratios between good and bad variables on $\boldsymbol{Z}^{(k)}_{good}$.

$$\log \frac{\boldsymbol{x}_j}{\boldsymbol{x}_l} = \boldsymbol{Z}^{(k)}_{good} \boldsymbol{\beta}^{(k)}_{(j,l);1} + \boldsymbol{\varepsilon}^{(k)}_{(j,l);1} \qquad \forall (j,l) \in J_{bad} \times J_{good}$$

We calculate the standardized residuals as in Equation (2.8) for each regression model

$$\boldsymbol{r}^{(k)}_{(j,l);1} = \frac{\hat{\boldsymbol{\varepsilon}}^{(k)}_{(j,l);1}}{\hat{\sigma}(\hat{\boldsymbol{\varepsilon}}^{(k)}_{(j,l);1})}, \tag{3.4}$$

where $\hat{\sigma}$ denotes the estimated standard deviation of the residuals.

6.2.2: regression analysis - bad/bad ratios. Furthermore we test, if the ratios between all bad variables deviate from the good ratios. Again only ratios containing new and meaningfull information are tested:

$$\log \frac{\boldsymbol{x}_j}{\boldsymbol{x}_l} = \boldsymbol{Z}^{(k)}_{good}\boldsymbol{\beta}^{(k)}_{(j,l);2} + \boldsymbol{\varepsilon}^{(k)}_{(j,l);2} \quad \forall j \in J_{bad} \forall l \in J_{bad} \setminus \{1, \dots, j\} \tag{3.5}$$

We again calculate standardized residuals as in Equation (2.8) for each regression model

$$\boldsymbol{r}^{(k)}_{(j,l);2} = \frac{\hat{\boldsymbol{\varepsilon}}^{(k)}_{(j,l);2}}{\hat{\sigma}(\hat{\boldsymbol{\varepsilon}}^{(k)}_{(j,l);2})},$$

where $\hat{\sigma}$ denotes the estimated standard deviation of the residuals.

6.2.3: robust distances within the space of good ratios. We also calculate robust distances $d^{(k)}_{1,i}$ for each observation within the space of good ratios $\boldsymbol{Z}^{(k)}_{good}$ using the *sign*-method (see Section 2.5.2).

6.2.4: robust distances within the space of bad ratios. Preceding Step 6.2.2 we also take a look at the robust distances of the "bad ratios". We therefore calculate the pairwise log-ratio matrix $\boldsymbol{Z}^{(k)}_{bad}$ as in Formula (3.3) between all bad variables $\boldsymbol{x}_j,\ j \in J_{bad}$. Utilizing the *sign*-method of Section 2.5.2 we obtain robust distances $d^{(k)}_{2,i}$ for each observation in $\boldsymbol{Z}^{(k)}_{bad}$.

It is very important to note, that (aside from degenerated settings, such as $p_{good} = 1 \land p_{bad} = 1$) there can occur certain subsets/outlier patterns $\boldsymbol{U}_k$, where not all steps of the ratio analysis in Step 6.2 are computeable:

- $p_{good} = 0$ or $p_{good} = 1$: If there is only one or even no good variable available, there are no pairwise log-ratios at hand to form $\boldsymbol{Z}^{(k)}_{good}$. The regressions in Steps 6.2.1 & 6.2.2, as well as the robust distance calculation within the good ratios in Step 6.2.3, can not be conducted. Only the bad ratios obtained in Step 6.2.4 may be checked for outliers.
- $p_{bad} = 0 \Rightarrow p_{good} = p_k$: If there are only good variables present in the current subset, regression analysis between bad/good (Step 6.2.1) and bad/bad (Step 6.2.2) variables is not applicable. Also due to the absence of bad ratios, Step 6.2.4 can not be performed. In this case

we only calculate robust distances between good ratios.

  – $p_{bad} = 1$: This case is similar to the previous one. However if only one variable gets flagged as an univariate outlier, we can test its ratios against all good variables in Step 6.2.1. Step 6.2.3 is of course applicable, Steps 6.2.2 & 6.2.4 may still not be performed.

6.3: outlier detection. Regarding the calculation of the outlier and distance flags we have to take into account, that the basis of our ratio analysis in Step 6.2 was the division of the variables into "good" and "bad" variables *within the subset/outlier pattern* $\boldsymbol{U}_k$. Therefore calculation and interpretation of outlier- and distance flags is only feasible for observations I_k (see Step 5) belonging to this particular subset/univariate outlier pattern.

6.3.1: outlier detection - bad/good ratios. We flag the ratio between the bad variable j and good variable l, $(j,l) \in J_{bad} \times J_{good}$ of subset $\boldsymbol{X}_k$ as an outlier according to

$$f^{(k)}_{(j,l);1;i} := \mathbb{1}(|r^{(k)}_{(j,l);1;i}| > 2.5),$$

for all observations $i \in I_k$ in $\boldsymbol{X}_k$.

6.3.2: outlier detection - bad/bad ratios. We flag the ratio between the bad variable j and bad variable l, $j \in J_{bad},\ l \in J_{bad} \backslash \{1, \dots, j\}$ of subset $\boldsymbol{X}_k$ as an outlier according to

$$f^{(k)}_{(j,l);2;i} := \mathbb{1}(|r^{(k)}_{(j,l);2;i}| > 2.5),$$

for all observations $i \in I_k$ in $\boldsymbol{X}_k$.

6.3.3: outlier detection - robust distances of good ratios. As stated in Section 2.5.1, we consider distances of observations in $\boldsymbol{Z}^{(k)}_{good}$ as outlying if

$$g^{(k)}_{1;i} := \mathbb{1}(d^{(k)}_{1;i} > \chi^2_{p^*;0.975})$$

for all observations $i \in I_k$. We use $p^* = ((p_{good} - 1) \cdot p_{good})/2$ degrees of freedom due to p_{good} "good" variables yield $\sum_{i=1}^{p_{good}-1} i = ((p_{good} - 1) \cdot p_{good})/2$ viable pairwise log-ratios in matrix $\boldsymbol{Z}^{(k)}_{good}$.

6.3.4: outlier detection - robust distances of bad ratios. As in Step 6.3.3, we consider distances of observations in $\boldsymbol{Z}^{(k)}_{bad}$ as outlying if

$$g^{(k)}_{2;i} := \mathbb{1}(d^{(k)}_{2;i} > \chi^2_{p^*;0.975})$$

for all observations $i \in I_k$. We use $p^* = ((p_{bad} - 1) \cdot p_{bad})/2$ degrees of freedom due to p_{good} "good" variables yield $\sum_{i=1}^{p_{bad}-1} i = ((p_{bad} - 1) \cdot p_{bad})/2$ viable pairwise log-ratios in matrix $\boldsymbol{Z}^{(k)}_{bad}$.

The special cases of subsets/outlier patterns, as described at the end of Step 6.2, and their impact on the ratio analysis of course carries over to the outlier detection step. Uncalculated standardized residuals and robust distances clearly can not be analyzed within this step.

6.4: interpretation. We now interpret the results of the ratio analyis and outlier detection obtained in Step 6.2 & 6.3 for subset $\boldsymbol{X}_k$. Note that as of now, all calculations and analyses were conducted on the pairwise log-ratios between the variables. In this step we try to combine the results of robust residuals and distances in order to make statements about which variable $\boldsymbol{x}_j$ leads to deviating ratios within the observations $i \in I_k$.

6.4.1: cellwise outlier. A cellwise outlier in $\boldsymbol{X}$ is a cell x_{ij} (whereas we only consider the applicable observations $i \in I_k$), whose ratio deviates from the good ratios of the data set. This is indicated by

$$f^{(k)}_{(j,l);1;i} = 1 \text{ and } g^{(k)}_{1;i} = 0, \qquad i \in I_k, (j,l) \in J_{bad} \times J_{good}.$$

If the robust distances $g^{(k)}_{1;i}$ between the good ratios in the above formula is 0, a large absolute standardized residual $r^{(k)}_{1;i}$ (obtained from the regression model (3.4)) is solely influenced by variable $\boldsymbol{x}_j$, i.e. cell x_{ij} in observation $i \in I_k$.

6.4.2: bivariate outlier. Following the intuition of Step 6.4.1, we also investigate the results of the regression models (3.5), where the bad ratios are considered. If the residual- and outlier flags indicate

$$f^{(k)}_{(j,l);2;i} = 1 \text{ and } g^{(k)}_{2;i} = 0, \qquad i \in I_k, j \in J_{bad},\ l \in J_{bad} \backslash \{1, \dots, j\}.$$

the large absolute standardized residual $r^{(k)}_{(j,l);2;i}$ (obtained from the regression model (3.5)) may only stem from the cells x_{ij} or x_{il}. We can not deduce, which one of those two leads to a deviating ratio, but we can acknowledge the bivariate nature of the deviating ratio.

6.4.3: multivariate outlier. Finally, we also check the computed robust distances. If for an observation $i \in I_k$

$$g^{(k)}_{1;i} = 1,$$

then the good ratios for this particular observation deviate from the main bulk of the good ratios of the other observations. We conclude that the cells $x_{ij}, j \in J_{good}$ of this observation are responsible for deviating ratios. Thus cells $x_{ij}, j \in J_{bad}$ cause deviating ratios, if

$$g^{(k)}_{2;i} = 1$$

for an observation $i \in I_k$.

Recapitulating, on completion of major Step 6, the algorithm has analyzed every outlier pattern/subset $\boldsymbol{X}_k$. Due to the observations of $\boldsymbol{X}$ being uniquely assigned to a subset, every row of $\boldsymbol{X}$ has now been analyzed on cellwise, bivariate and multivariate outliers.

3.3.2 Advantages and limitations of the algorithm

For the detection of cellwise outliers the major disadvantage of the two algorithms presented so far was the usage of the isometric log-ratio transformation used for expressing the ratios between variables. Due to the transformation always including other variables, the new coordinates are heavily influenced in the presence of outliers. Therefore detecting the cellwise source of deviating ratios has proven to be quite difficult so far. This approach focuses on investigating the pairwise log-ratios between good and bad variables. Due to the intelligent use and combination of regression analysis and robust distance calculation, this approach is able to detect univariate, bivariate and multivariate outliers in the original sample space. It even uses all observations to estimate the regression hyperplane as well as the center and covariance structure of the data cloud, making those estimates quite potent. However one major point of criticism comes with the initial separation into good and bad variables. This segmentation is based on a univariate outlier detection on the standardized absolute values of the variables. In the sense of compositional data, one can think of examples were deviating absolute values are not necessarily responsible for deviating ratios.

3.4 Detect deviating cells on pairwise log-ratios

Following our previous approach, we continue to utilize the pairwise log-ratio matrix as in Formula (3.3) and search for deviating ratios resulting from outlying cells. Rousseeuw and Van den Bossche [2016] propose a method (*detectDeviatingCells*) to detect cellwise outliers in the data and take the correlations between the variables into account. It furthermore has no restriction on the number of contaminated rows and can deal with high dimension, which is an important property to have, due to the possibly large number of pairwise log-ratio combinations.

The main idea of the algorithm is to first impute the intial data set and compute the pairwise log-ratio matrix between the variables. We now apply the *detectDeviatingCells* method on the newly obtained data set. The method starts by standardizing the obtained matrix and flagging the cells that stand out in their column. Next, each data cell is estimated based on the unflagged cells in the same row whose column is correlated with the column in question. Finally, a cell for which the observed value differs much from its estimated value is considered anomalous. The output of the procedure is now used to make statements about the outlyingness of each cell in the initial data set by combining the results of each ratio a particular variable is involved in.

3.4.1 Detailed description of the algorithm

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ be the $n \times p$ dimensional compositional data set.

Step 1: imputation. We deal with structural zeros within the compositional data by setting them to missing beforehand. Afterwards $\boldsymbol{X}$ is imputed using robust imputation methods resulting in $\boldsymbol{X}^{imp}$.

Step 2: compute pairwise log-ratio matrix. Compute the $n \times ((p-1) \cdot p)/2$ dimensional pairwise log-ratio matrix $\boldsymbol{Y}$ as in Formula (3.3) between all columns of $\boldsymbol{X}^{imp}$. We then continue with the approach outlined in Rousseeuw and Van den Bossche [2016] to detect deviating cells within the pairwise log-ratio matrix. The algorithm proceeds as follows.

Step 3: standardization. For each column $\boldsymbol{y}_j,\ j = 1, \dots, p$ of $\boldsymbol{Y}$ we estimate

$$m_j = robLoc_i(y_{ij}) \text{ and } s_j = robScale_i(y_{ij} - m_j), \tag{3.6}$$

where *robLoc* is a robust estimator of location (as defined in Equation (2.10)) and *robScale* is a robust estimator of scale (as defined in Equation (2.11)), which assumes its argument has already been centered. We then standardize $\boldsymbol{Y}$ to $\boldsymbol{Z}$ by

$$z_{ij} = \frac{y_{ij} - m_j}{s_j}. \tag{3.7}$$

Step 3: apply univariate outlier detection to all variables. After the column-wise standardization in (3.7) we define a new matrix $\boldsymbol{U}$ with entries $u_{ij} = z_{ij}$ except when

$$|z_{ij}| > c \tag{3.8}$$

in which case we set u_{ij} to missing. Note that (3.8) only uses variable j itself, so it is purely column-wise. The cutoff value c is taken as

$$c = \sqrt{\chi^2_{1,p}}, \tag{3.9}$$

where $\chi^2_{i,p}$ is the p-th quantile of the chi-squared distribution with 1 degree of freedom, where the probability p is often chosen as 99% so that under ideal circumstances only 1% of the entries get flagged.

Step 4: bivariate relations. For any two variable $h \neq j$ we compute their correlation as

$$cor_{jh} = robCorr_i(u_{ij}, u_{ih}), \tag{3.10}$$

where *robCorr* is a robust correlation measure given in Appendix B Equation (1). From here onward we will only use the relation between variables j and h when

$$corr_{jh} \geq corrlim \tag{3.11}$$

in which *corrlim* is set to 0.5 by default. Variables j that satisfy (3.11) for some $h \neq j$ will be called *connected*. The others are called *standalone* variables. For the paris (j, h) satisfying (3.11) we also compute

$$b_{jh} = robSlope_i(u_{ij}|u_{ih}),$$

where *robSlope* computes the slope of a robust regression line without intercept that predicts variable j from variable h (see Appendix B).

Step 5: estimated values. Next we compute estimated values $\hat{z}_{ij}$ for all cells. For each variable j we consider the set H_j consisting of all variables h satisfying (3.11), including j itself. For all $i = 1, \ldots, n$ we then set

$$\hat{z}_{ij} = G(\{b_{jh}u_{ih} \mid h \in H_j\}), \tag{3.12}$$

where G is a combination rule applied to these numbers, which omits the `NA` values and is zero when no values remain. Our current preference for G is a weighted mean with weights $w_{jh} = |cor_{jh}|$ but other choices are possible, such as a weighted median.

Step 6: deshrinkage. Note that a prediction such as in Equation (3.12) tends to shrink the scale of the entries, which is undesirable. We could try to shrink less in the individual terms $b_{jh}u_{ih}$ but this would not suffice because these terms can have different signs for different h. Therefore, we propose to deal with the shrinkage after applying the combination rule in Equation (3.12). For this purpose we replace $\hat{z}_{ij}$ by

$$\hat{z}_{ij} \cdot robSlope_{i'}(z_{i'j}|\hat{z}_{i'j})$$

for all i and j.

Step 7: flagging cellwise outliers in the pairwise log-ratio matrix. In Steps 5 and 6 we have computed estimated values $\hat{z}_{ij}$ for all cells. Next we compute the standardized cell residuals

$$r_{ij} = \frac{z_{ij} - \hat{z}_{ij}}{robScale_{i'}(z_{i'j} - \hat{z}_{i'j})}.$$

In each column j we then flag all cells with

$$f_{ij} = \mathbb{1}(|r_{ij}| > c) \tag{3.13}$$

as anomalous.

Step 8: flagging rowwise outliers in the pairwise log-ratio matrix. In order to decide whether to flag row i we could just count the number of cells whose $|r_{ij}|$ exceeds a cutoff value a, but this would miss rows with many fairly large $|r_{ij}| < a$. The other extreme would be to compare $ave_j(r_{ij}^2)$ to a cutoff, but then a row with one very outlying cell would be flagged as outlying, which would defeat the purpose. We choose an approach in between. Under the null hypothesis of multivariate gaussian data without any outliers, the distribution of the r_{ij} is close to standard gaussian, so the cdf of r_{ij}^2 is approximately the cdf F of χ_1^2. This leads us to the criterion

$$T_i = ave_{j=1}^{p} F(r_{ij}^2) - \frac{1}{2}$$

which lies between -0.5 and 0.5 . We then standardize the T_i as in Formula (3.7) and flag the rows i for which the standardized T_i exceed the cutoff value c of Formula (3.11).

The algorithm in Rousseeuw and Van den Bossche [2016] still proceeds, however after this Step we achieved to flag cell- and rowwise outlier in our initial matrix of pairwise log-ratios $\boldsymbol{Y}$.

Step 9: deduce cellwise outliers in the initial data set $\boldsymbol{X}$. For each column p in $\boldsymbol{X}$ we now check the corresponding detected anomalous data cells in the pairwise log-ratio matrix $\boldsymbol{Y}$ detected by the algorithm. Let J_j^{pair} be the set of column indices of the pairwise log-ratios in $\boldsymbol{Y}$ including variable $\boldsymbol{x}_j$, $\#J_j^{pair} = p - 1$. We then can summarize the cellwise flags obtained for these ratios

$$\sum_{j' \in J_j^{pair}} f_{ij'} \quad \text{for each observation } i.$$

We further conclude, that if for example more than 50% of the ratios corresponding to variable $\boldsymbol{x}_j$ are flagged as outliers for observation i, cell x_{ij} is responsible for the deviating behavior of the ratios within this observation. Applying this rule for all variables $\boldsymbol{x}_j$ we obtain cellwise flags g_{ij} for each data point x_{ij}.

Step 10: deduce rowwise outliers in the initial data set $\boldsymbol{X}$. In Step 8 we already received rowwise flags T_i for outlying rows i in the log-ratio matrix $\boldsymbol{Y}$. We conclude that outlying rows in the ratios of $\boldsymbol{Y}$ stem from the corresponding outlying row in $\boldsymbol{X}$. Therefore we take the obtained flags T_i one-to-one to flag rows in $\boldsymbol{X}$. Furthermore as in previous approaches outlier flags g_{ij} obtained in Step 9 may be summed up

$$\sum_{i=1}^{p} g_{ij} \quad \text{for the } i\text{-th observation,}$$

indicating how often this observation was identified as an outlier. Based on this sum the whole row may be flagged as an outlier.

3.4.2 Advantages and limitations of the algorithm

In this approach we once more focus on the pairwise log-ratio matrix between the compositional variables. Here the major advantage lies in the usage of the *detect-DeviatingCells* algorithm, which detects anomalous data cells within the log-ratios. Here this algorithm starts with a univariate outlier detection as well, although this time the initial step is executed on the ratios, which is in the interest of compositional data analysis. The crucial point when using this algorithm is however again the interpretation of the results produced by *detectDeviatingCells*. Anomalous data cells are now flagged on the matrix of the pairwise log-ratios. In order to find cells, which lead to those deviating ratios, we have to consider all pairwise log-ratios within the observation containing the corresponding cell. One has to find a good rule of thumb concerning how many pairwise log-ratios have to deviate in order to flag the cell in the original matrix as an outlier.

Chapter 4

Application to household expenditure data

After the thorough discussion of our algorithms, we now apply them on actual data and review the results. We consider a data set of household expenditures from the year 2008 of Albania (see Hron et al. [2015]) provided by the World Bank[1]. The data were obtained through a survey, where participants were asked about their household consumption over a given period of time in various spending categories. These categories range from different kinds of food-products over general living expenditures like gas, electricity or water to expenses for education, health and others. The type and number of categories vary from survey to survey but have in common, that the combined categories reflect the whole consumption of a household for this particular time frame. The Albanian household survey consists of 3600 households, including 14785 individuals. We are going to analyze the seven major parts, namely household consumption on

- food and non-alcoholic beverages (*food*),
- alcoholic beverages, tobacco and narcotics (*alcohol*),
- clothing and footwear (*clothing*),
- housing, water, electricity, gas and other fuels (*housing*),
- communication (*commun.*),
- education (*education*) and
- miscellaneous goods and services (*misc.*).

The amount of zeros in the data varies between close to 0% up to nearly 10% per variable. Out of the total number of households, 2903 observations do not include any zeros, 260 observations have zeros in variable alcohol only, 250 observations contain zeros only in variable clothing. For a full overview of the data set's zero structure see Figure 4-1. The figure indicates the amount of zeros per variable on the left, whereas

[1]http://datatopics.worldbank.org/consumption/

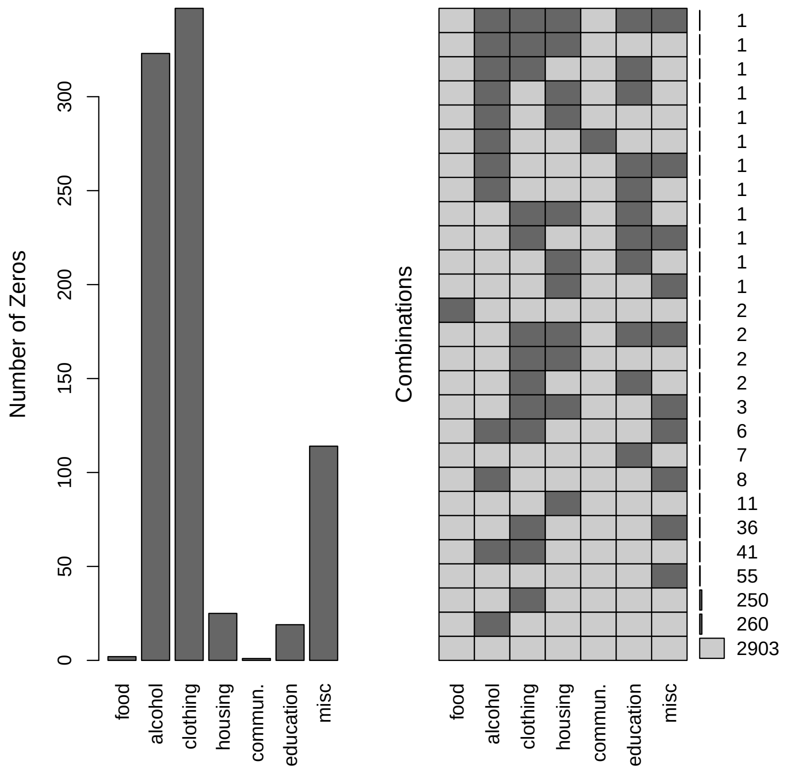

Figure 4-1: Amount of zeros per variable (left) and zero patterns (right) for the Albanian consumption data.

the right figure depicts the amount of zeros per zero pattern. In addition to household expenditure variables, the data set provides additional explanatory variables, which can be used in the regression analysis of the imputation and subset approach. Table 4-1 gives further insight into those variables.

Name	Label	Notes
hhsize	Household size	Number of household members (based on country-specific definition of a household). Does not include paying boarders, domestic servants, and visitors.
adeq_fao	Adults equivalent (FAO scale)	Number of adult equivalent in the household, computed based on the standard FAO scale. The variable is calculated for each household by summing up the following adult equivalent factor given to each member according to his/her age and sex: Male Female <1 yr 0.27 0.27 1-3 yrs 0.45 0.45 4-6 yrs 0.61 0.61 7-9 yrs 0.73 0.73 10-12 yrs 0.86 0.78 13-15 yrs 0.96 0.83 16-19 yrs 1.02 0.77 20 and above 1.00 0.73
m_00_15	Nb of males, 0 to 15 years	Number of male household members aged 0 to 15 years. Undefined age are counted in "16 to 59"
m_16_59	Nb of males, 16 to 59 years	Number of male household members aged 16 to 59 years. Undefined age are counted in "16 to 59"
m_60p	Nb of males, 60 years and over	Number of male household members aged 60 years and over. Undefined age are counted in "16 to 59"
f_00_15	Nb of females, 0 to 15 years	Number of female household members aged 0 to 15 years. Undefined age are counted in "16 to 59"
f_16_59	Nb of females, 16 to 59 years	Number of female household members aged 16 to 59 years. Undefined age are counted in "16 to 59"

f_60p	Nb of females, 60 years and over	Number of female household members aged 60 years and over. Undefined age are counted in "16 to 59"
hhagey	Age of household head	Age (in years) of the head of household. Each household, for the purposes of this data set, has one and only one head. The head of the household is the member declared as such by the respondent(s). In cases where more than one head is identified, the older one is considered as head.

Table 4-1: Additional explanatory variables provided by the data set, which can be used in the regression analysis of the imputation and subset approach.

The approaches presented in Chapter 3 are now applied on the consumption data and the additional variables. Here an MM-estimator (see Section 2.3.2) for regression analysis and the k-Nearest Neighbor Imputation (see Appendix A) to impute missing values are used. For the following discussion we use the notation

– *imputation* for the imputation approach,

– *subset* for the subset approach,

– *pairlog* for the pairwise log-ratio approach,

– *pairlogr* for our modified algorithm from Rousseeuw and Van den Bossche [2016] to detect deviating cells applied on the pairwise log-ratios.

Table 4-2 summarizes the basic results of the approaches. As expected the imputation

Number of cellwise outliers detected by approach and variable:

	food	alcohol	clothing	housing	commun.	education	misc	sum
imputation	65	84	55	119	95	113	90	621
subset	63	87	53	118	104	127	96	648
pairlog	39	19	15	64	43	18	55	253
pairlogr	58	59	36	100	75	48	75	451

Number of rowwise outliers detected by approach:

imputation	subset	pairlog	pairlogr
301	82	160	28

Table 4-2: Summary of cellwise and rowwise outliers detected by approach and variable.

and subset approach show similar results when it comes to the total amount of cellwise

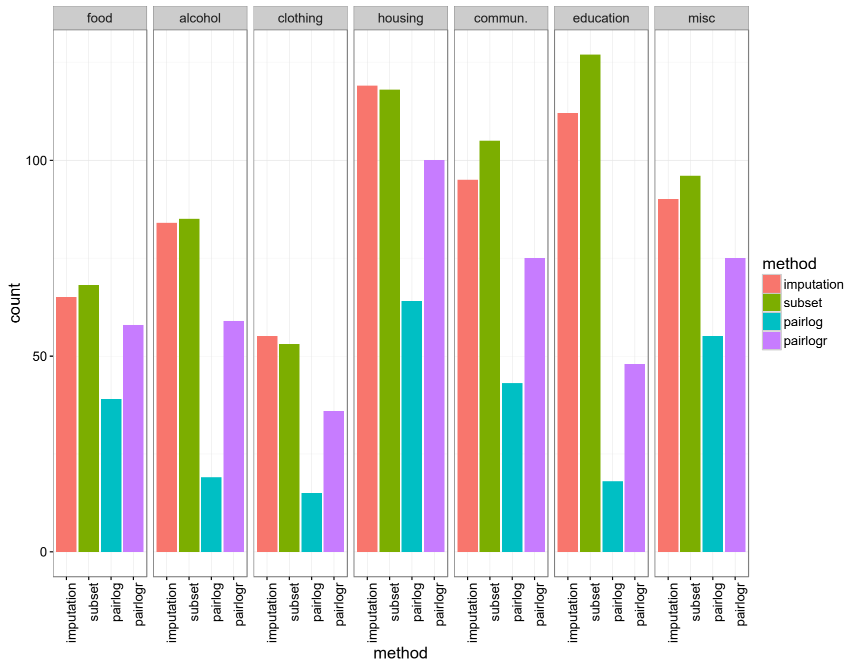

Figure 4-2: Number of cellwise outliers detected by variable and approach

outliers detected. Only the pairlog approach seems to detect significantly less than the other approaches. Concerning the distribution between variables, Figure 4-2 also shows that especially variables *housing, communication* and *education* seem to be responsible for deviating ratios within observations. The imputation approach also tends to detect the most rowwise outliers, although the subset approach does not mimic this behavior this time. Closer inspection showed that this discrepancy is due to the different robust distance calculations used in both approaches. These lead to different rowwise results when applying the sum rules in Step 4.2 (3.1) and Step 5.2 (3.2) of the imputation and subset approach respectively. As stated in Step 8 of the pairlogr approach, the authors of the *detectDeviatingCells* algorithm seem to be very cautious when marking rowwise outliers. This may explain the relatively low amount of detected outlying observations.

Following this initial investigation we now want to compare the results on a cellwise level. To do this Figure 4-3 provides a cell map of the expenditure data set. The cell map shows a plot of every data cell. Cells are visualized by color-coded squares indicating

- 🟨 regular cells,
- 🟥 cellwise outliers
- 🟦 bivariate cellwise outliers (only detected by the *pairlog* approach, see Step 6.4.2)
- ⬛ rowwise outliers,
- ⬜ missing values.

Due to the data set containing 3600 observations we focus on a subset that contains all cell- and rowwise scenarios that may be detected by our algorithms. On a positive note the algorithms seem to detect mostly the same outliers, wheres the results from similar algorithms appear to agree even more. The cell map however also provides room for interpreting the advantages and disadvantages of the different algorithms. To give an example, in observation 1727 the pairlog approach detects a bivariate outlier in variables *commun.* and *misc.*. The pairlogr approach, which is also based on the pairwise log-ratio matrix between the variables, supports this proposition. The other two approaches also detect a cellwise outlier for observation 1727 and variable *misc.*. However a cellwise outlier in *commun.* remains undetected by both imputation and subset approach. This is probably due to the isometric log-ratio transformation used in these approaches; the cellwise outlier in *misc.* possibly masks the outlier in variable *commun.*. Furthermore it is very interesting, that the imputation approach detects a cellwise outlier in observation 1668 whereas the subset approach marks the whole observation as an outlier. Here we can recognize the presence of a missing value. Recapitulating that the imputation and subset approach treat missing information in a different way, one can imagine that the subset approach comes to a different result when analyzing this particular zero pattern.

We can further analyze the consumption data by looking at the sorted data set. Figure 4-4 shows the results of each of the four approaches, although each plot is sorted by a different variable in ascending or descending order. Due to the lack of space only the top or bottom 100 observations are shown. The *imputation* approach in Figure 4-4 shows the top 100 observations ordered by variable *food* in descending order. Combining the cellwise information with the actual data values we can observe that outlying cells in variable *food* often come along with high, outlying values in variables *clothing, housing, misc.*. Therefore these observations probably represent the affluent part of society, which may also explain the deviating compositional structure of the observations. The *subset* approach shows variable *alcohol* in descending order. With Albania being a Muslim-majority country [Department, 2014] we have an overall narrow value range in variable *alcohol.* Thus it does make sense that higher expenditures within this category lead to deviating ratios within the affected observations. This behavior seems to get detected by algorithm quite well. Especially observations with high or low values in variables *housing* and *education*, see the *pairlog* and *pairlogr* approach in Figure 4-4, best imply the gap between rich and poor in Albania. Cellwise outlier flags back up the thesis, that these observations differ from the majority of the data. Investigating the underlying data, outstanding values in *housing* and/or *education* in general go with high expenditures in *food*, *clothing* and *communication*, whereas outliers on the lower end of the scale (note that in the *pairlogr* plot the data set is orderd by variable *education* ascending) show the exact opposite.

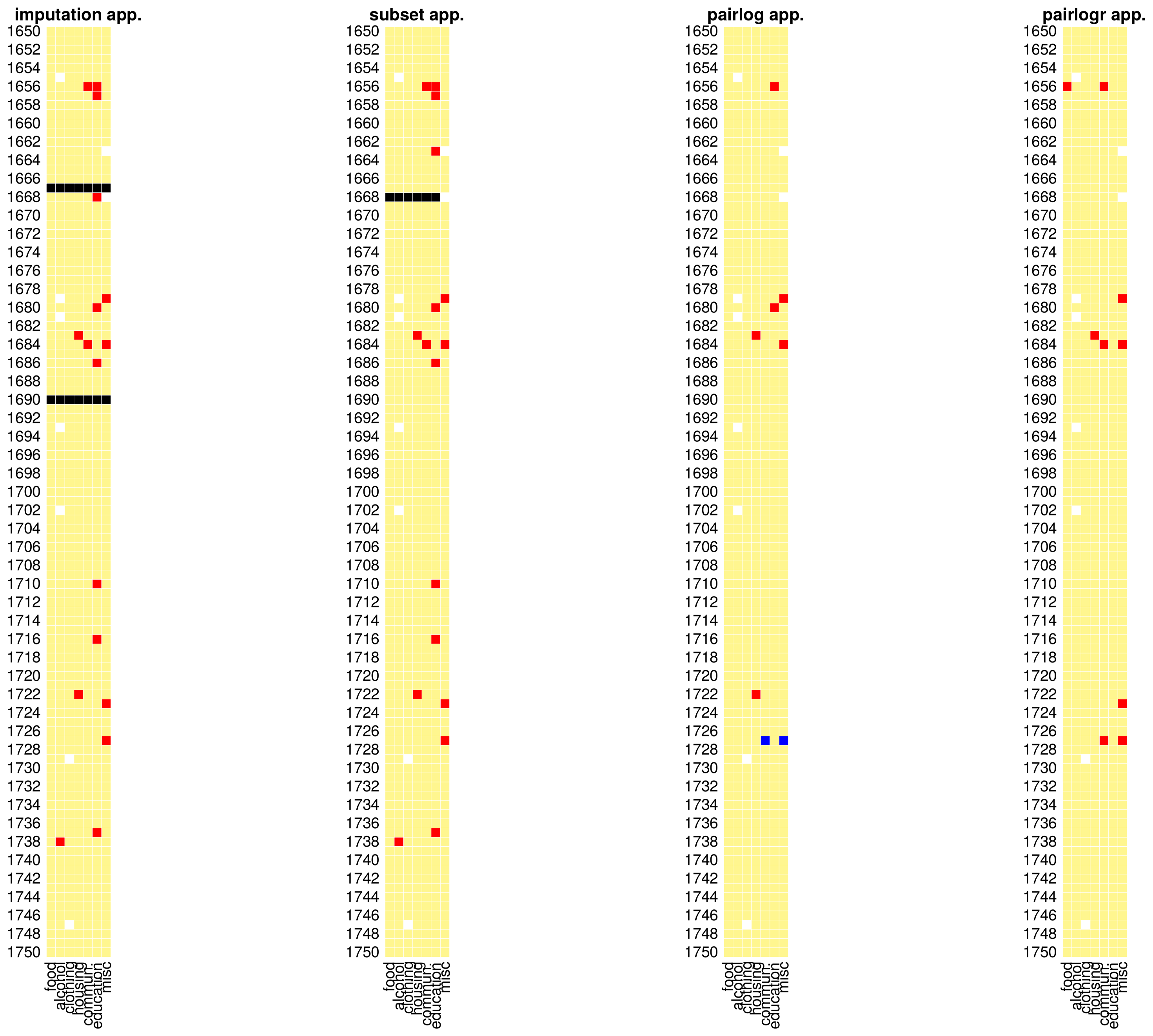

Figure 4-3: Partial cell map for observations 1650-1750 of the Albanian consumption data.

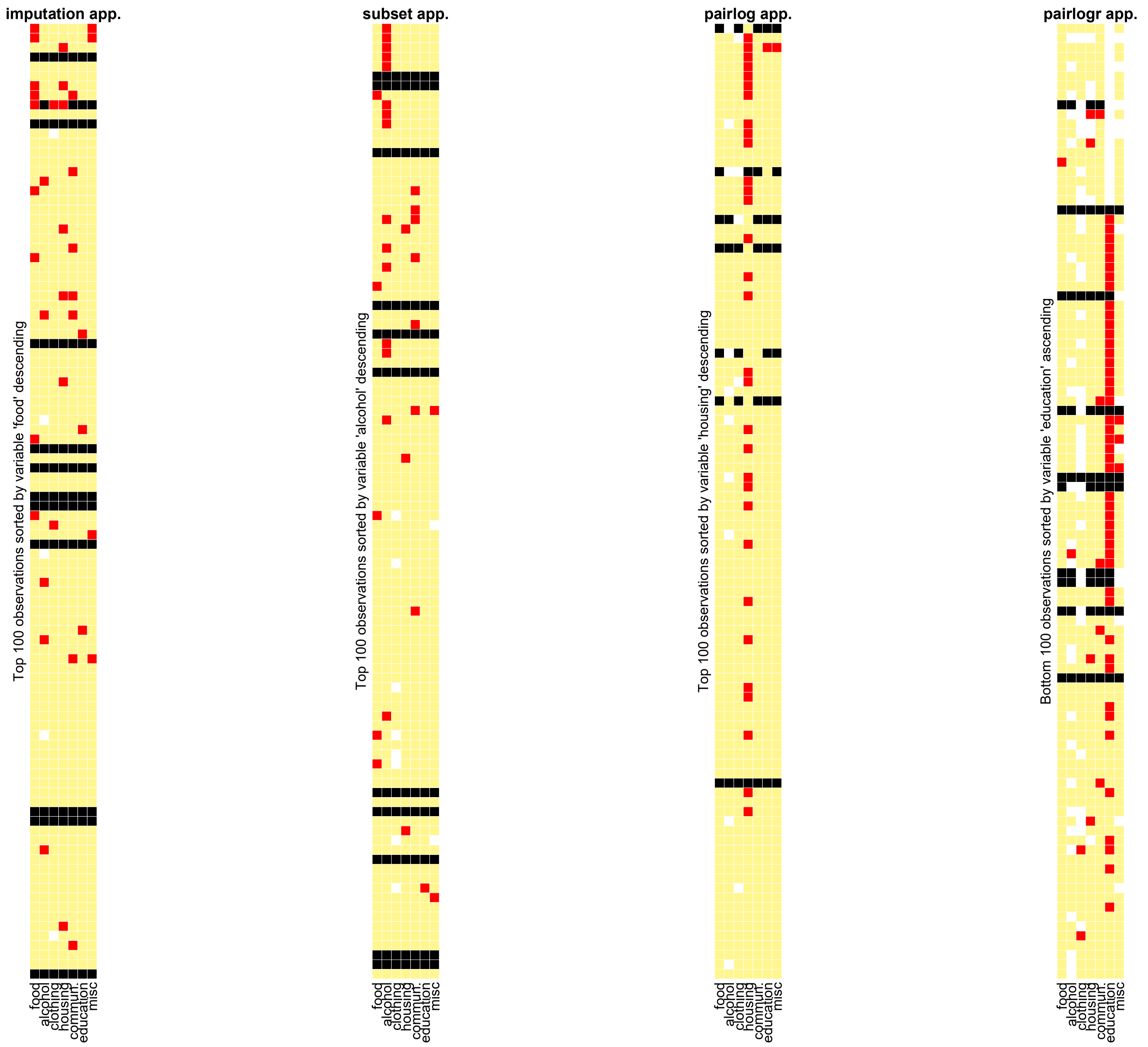

Figure 4-4: Partial cell map for sorted observations of the Albanian consumption data. Data is always displayed top down, meaning the minimum or maximum of the variable in question is always featured within the very top observation.

Chapter 5

Simulation study

We now want to illustrate the behavior of the presented approaches in the presence of outliers and missing values. The algorithms are evaluated on generated synthetic data sets based on the real household expenditure data set. The rowwise outlier information obtained in Chapter 4, as well as randomly adding cellwise outliers, are used to achieve appropriate contaminated compositional data sets.

5.1 Simulation of data

Let $\boldsymbol{X}^{orig} = (\boldsymbol{x}_1^{orig}, \ldots, \boldsymbol{x}_p^{orig})$ be the compositional and $\boldsymbol{Y}^{orig} = (\boldsymbol{y}_1^{orig}, \ldots, \boldsymbol{y}_D^{orig})$ the non-compositional variables of the household expenditure data $(\boldsymbol{X}^{orig}, \boldsymbol{Y}^{orig})$ from Chapter 4. Synthetic data are then generated as follows:

Step 1: imputation. Impute $\boldsymbol{X}^{orig}$ to obtain $\boldsymbol{X}^{imp}$.

Step 2: modeling additional variables. Estimate an additional variable $\boldsymbol{y}_i^{orig}$ using a generalized linear poisson-model $\boldsymbol{y}_i^{orig} \sim \boldsymbol{X}^{imp}$ for all $i = 1, \ldots, D$. In particular, the simulation studies below use the additional variables

- *hhsize*,
- *m_00_15*,
- *m_16_59*,
- *m_60p*,
- *f_00_15*,
- *f_16_59* and
- *f_60p*,

because they best follow a poisson distribution.

Step 3: parameter estimation. Represent the compositional data $\boldsymbol{X}^{orig}$ in the new coordinates using the isometric log-ratio transformation $\boldsymbol{Z}^{orig} = ilr(\boldsymbol{X}^{imp})$. The rowwise outlier information on the original data set obtained in Chapter

4 is then used to extract outlying observations $\boldsymbol{Z}^{orig\_out}$. We now estimate the covariance structure of this set and choose the following additional parameters:

$$\begin{aligned}\boldsymbol{\mu}_0 &= (1,1,1,1,1,1)'\\ \boldsymbol{\Sigma}_0 &= diag(1,1,1,1,1,1)\\ \boldsymbol{\mu}_1 &= (-6,6,-6,6,-6,6)'\\ \boldsymbol{\Sigma}_1 &= cov(\boldsymbol{Z}^{orig\_out})\end{aligned}$$

This results in the parameters $(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ for the location and covariance of the regular and outlying observations in the new coordinates. $\boldsymbol{\mu}_1$ is used to further differentiate "bad" data points.

Step 4: data generation and contamination. When it comes to data generation and contamination we consider the contamination rate ε. Based on ε we generate equal amounts of

– rowwise outlier,

– univariate cellwise outlier and

– bivariate cellwise outlier,

each of these with a contamination rate of $\varepsilon/3$, ensuring that ε amount of the data is contaminated.

4.1: rowwise contamination. Therefore we first draw a synthetic data set from the distribution

$$\boldsymbol{Z}^{synth} \sim \mathcal{F},$$

where $\mathcal{F} = (1-\varepsilon/3)\mathcal{F}_0 + (\varepsilon/3)\mathcal{F}_1$. We choose $\mathcal{F}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\mathcal{F}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$.

4.2: cellwise contamination. Concerning contaminating individual or pairs of cells we first back-transform the obtained data set $\boldsymbol{Z}^{synth}$ using the inverse log-ratio transformation as in Formula (2.7)

$$\boldsymbol{X}^{synth} = invilr(\boldsymbol{Z}^{synth}).$$

We then contaminate $\varepsilon/3$ of the uncontaminated rows in $\boldsymbol{X}^{synth}$ with a single cellwise outlier by adding a to a cell. Moreover we contaminate $\varepsilon/3$ uncontaminated rows with a double cellwise outlier by adding a, b to two particular cells.

Step 5: estimating additional variables. Corresponding additional variables $\boldsymbol{Y}^{synth}$ can be produced by using the regression models obtained in Step 2. This is done by predicting the response variable $\boldsymbol{y}_i^{synth}$ with the new data set $\boldsymbol{X}^{synth}$.

Step 6: adding zeros. Concluding zeros are inserted into the obtained set $\boldsymbol{X}^{synth}$. In order to receive a meaningful zero structure for the subset approach, zeros are spread in only two variables with a zero rate of λ per variable.

The whole procedure generates synthetic data sets $(\boldsymbol{X}^{synth}, \boldsymbol{Y}^{synth})$, which copy the compositional behavior of the original expenditure data as well as providing feasible additional variables.

5.2 Simulation setup

Based on this set-up, two experiments are performed. In the first case the fraction of outliers ε is fixed at 0.1 and the zero rate λ is varied from 0.0 to 0.3 by 0.025. Alternatively, in the second case the zero rate λ is fixed at 0.1 and the fraction of outliers ε is varied from 0.0 to 0.3 by 0.025. For each configuration $m = 50$ data sets of dimension $n = 500$, $p = 7, D = 7$ are generated and for each of them the outliers are identified using all four approaches. To evaluate the results we consider the following four error measures:

- FN.row – Average outlier error rate (rowwise): the average percentage of rowwise outliers that were not identified – false negatives or masked outliers.
- FN.cell – Average outlier error rate (cellwise): the average percentage of cellwise outliers that were not identified – false negatives or masked outliers.
- FP.row – Average non-outlier error rate (rowwise): the average percentage of non-rowwise outliers that were classified as outliers – false positives or swamped non-outliers.
- FP.cell – Average non-outlier error rate (rowwise): the average percentage of non-rowwise outliers that were classified as outliers – false positives or swamped non-outliers.

5.3 Simulation results

The results of the simulation are presented in Figure 5-1 and Figure 5-2. Recapitulating we have introduced two ways of dealing with missing values and structural zeros. The first one imputes missing information, the second one divides the data set into subsets according to the zero patterns. Robust imputation is used by the *imputation, pairlog* and *pairlogr* approach, whereas the division into zero patterns is used by the *subset* approach.

5.3.1 Results for varying the fraction of missing values

For a fixed outlier rate of 0.1 and an increasing fraction of missing values, Figure 5-1 shows that the *subset* approach sometimes fails to detect outliers resulting in an

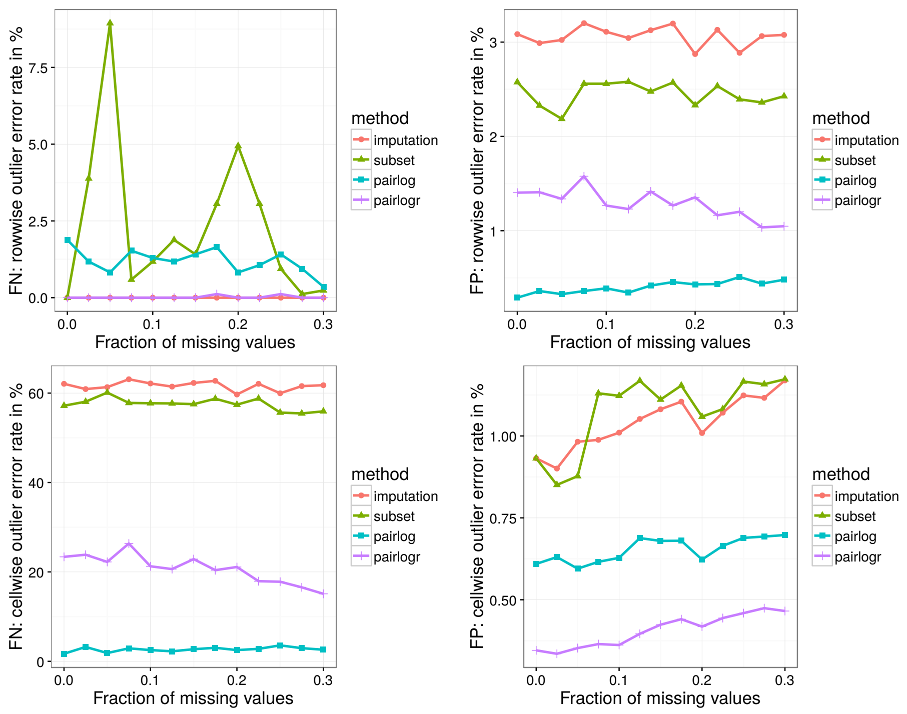

Figure 5-1: Average outlier error rate (left) and average non-outlier error rate (right) for fixed fraction of 10% outliers and varying percentage of zeros.

increased FN-rate for rowwise outliers. As stated in Section 3.2.2 the subset approach comes with the downside, that some subsets turn out to be too small for the regression- and distance based methods used in the algorithm. Therefore outliers within those subsets may not be detected, possibly resulting in the spikes within the FN-rate for certain simulation runs. All other approaches are able to work on the full, imputed data sets and hence seem to detect rowwise outliers quite well. The FP-rate for rowwise outliers shows that especially the *imputation* and *subset* approach detect too many outliers. Both of these approaches use an isometric log-ratio transformation to analyze the ratios between the parts of an observation. As discussed in Section 3.1.2 and 3.2.2, due to the nature of the isometric log-ratio transformation, observations represented in the new coordinates are almost always influenced by possible multivariate outliers in the original sample space. Since our experimental setup includes single and double cellwise contamination, this contamination may spread within the new coordinates after transformation and thus leading the *imputation* and *subset* approach to a rowwise outlier identification and overall higher FP-rates. This particular behavior also manifests in the results for cellwise outlier in Figure 5-1. The isometric log-ratio transformation makes it really hard to detect cellwise contamination, leading to high amounts of undetected outliers in the *imputation* and *subset* approach. The *pairlog* approach excels in this contamination setup resulting in very low FN-rates and quite good FP-rates. The *pairlogr* approach holds the middle ground, even resulting in slightly better cellwise FP-rates compared to the *pairlog* approach.

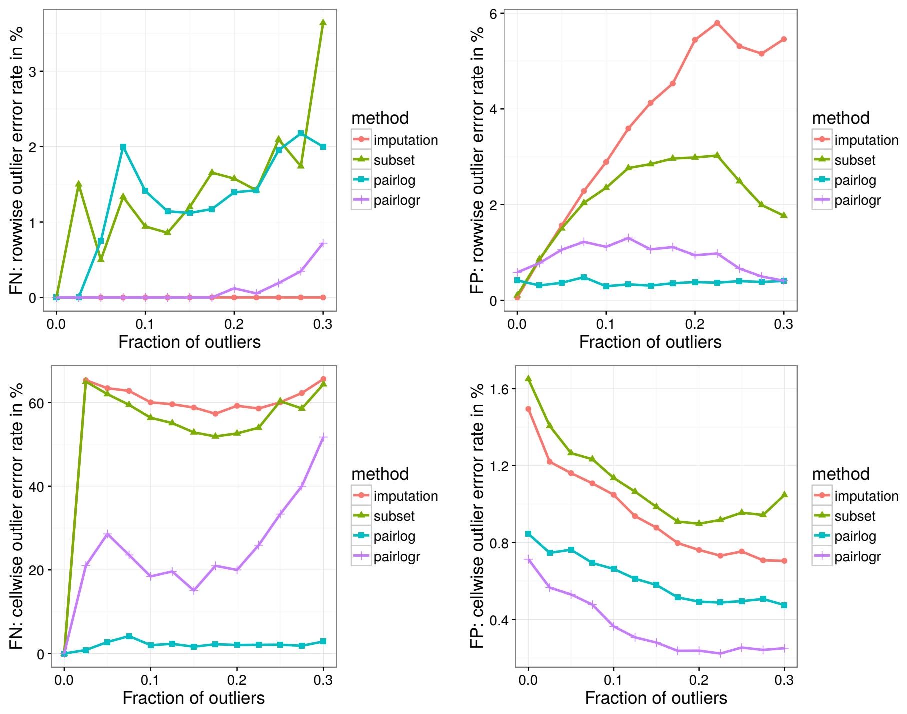

Figure 5-2: Average outlier error rate (left) and average non-outlier error rate (right) for fixed fraction of 10% zeros and varying fraction of rowwise (top)/cellwise (bottom) outliers.

5.3.2 Results for varying the fraction of outliers

Figure 5-2 shows the behavior of the algorithms for a fixed fraction of missing values of 0.1 and an increasing outlier rate. When it comes to rowwise outlier detection all algorithms seem to produce quite good results over the course of the simulation run, all of them having a FP-rate below 2% for an outlier rate between 0 and 0.2 and slightly increasing towards the end. Concerning the FP-rate we observe that especially the *imputation* method performs significantly worse than the other approaches for increasing outlier rates, even when compared to the *subset* approach. This may be due to the *imputation* approach being the only approach using robust squared mahalanobis distances instead of the *sign*-method, possibly resulting in different distance flags and thus different results. As discussed above an overall higher FP-rate is to be expected when using the *imputation* and *subset* approach due to the usage of an isometric log-ratio transformation. In general the *pairlog* and *pairlogr* approach seem to produce quite stable results, even for an increasing amount of rowwise outliers. Concerning cellwise outliers Figure 5-2 confirms the issues that seem to come along with the usage of an isometric log-ratio transformation. Throughout the whole simulation study the *imputation* and *subset* approach fails to detect a major amount of cellwise outliers with an FN-rate between 50% and 60%. Overall the *pairlog* approach produces very good results with an FN-rate of under 5% and an FP-rate of under 1% even for large fractions of outliers. Again the *pairlogr* holds the middle ground when it comes to cellwise outlier detection, even producing slightly better than the *pairlog* approach in terms of FP-rate.

Chapter 6

Summary and Conclusio

The analysis of compositional data is based on working in coordinates (using e.g. the isometric log-ratio transformation) or analysing pairwise log-ratios. Zeros in the parts thus cause severe difficulties for the analysis. Log-ratio transformations represent the compositional information into new coordinates. Outliers within these coordinates may be detected, however it remains unclear which particular parts of the composition led to the deviating ratios in question. Therefore this thesis presented four exploratory tools for identifying cellwise outliers in compositional data with structural zeros. The *imputation* approach deals with structural zeros by imputing them using robust imputation methods. Imputation is done in a way, that no additional outliers are generated when filling missing information. We can therefore use the full information without worrying about zeros or additional outliers. The compositional variables are expressed into new coordinates using an isometric log-ratio transformation. Afterwards a combination of robust regression and robust distance calculation is applied on the data set to detect row- and cell-wise outliers. The regression approach also enables an easy integration of external, non-compositional variables (e.g. demographic information), possibly gaining valuable information.

The *subset* approach follows this proposal, however missing values are handled by dividing the data set into unique subsets (patterns) according to the zero-pattern occurring in each row. Therefore imputation is not needed and the process of representing the variables in the new coordinates using an isometric log-ratio transformation and then performing cellwise regression and distance calculation can be executed on each subset. One disadvantage of this approach is that subsets may turn out to be too small for our outlier detection methods. Potential observations within those subsets, that deviate from the main bulk of the data, can not be detected. Furthermore due to both methods using the isometric log-ratio transformation, cellwise outlier detection proves to be very difficult. Due to the nature of the transformation, observations represented in the new coordinates are almost always influenced by possible multivariate outliers in the original sample space. This makes it quite difficult to deduce, which cell/cells of the observation in the untransformed space leads/lead to outlying ratios.

These difficulties led us to the *pairlog* and *pairlogr* algorithms, which focus on investigating the pairwise log-ratios between the variables. Both approaches work on the imputed data set. The *pairlog* method introduces the concept of “good” and "bad" variables and then uses clever combination of regression analysis and robust distance calculation to detect row- and cell-wise outliers in the original sample space. The major point of criticism with this approach however comes with the initial separation into good and bad variables. This segmentation is based on an univariate outlier detection on the standardized absolute values of the variables. In the sense of compositional data, one can think of examples where deviating absolute values are not necessarily responsible for deviating ratios.

Again, utilizing the concept of pairwise log-ratios between variables, the *pairlogr* approach applies the *detectDeviatingCells* method proposed by Rousseeuw and Van den Bossche [2016] on the pairwise log-ratio matrix to detect cellwise outliers in the data. This method takes the correlation between the variables into account, has no restriction on the number of contaminated rows, and can deal with high dimensions. This property is of significant importance, due to the possibility of large numbers of pairwise log-ratio combinations. Although this algorithm also starts with an univariate outlier detection, this time the initial step is executed on the ratios, which is in the interest of compositional data analysis. The crucial point is however the interpretation of the results produced by *detectDeviatingCells*. Anomalous data cells are flagged on the matrix of pairwise log-ratios. In order to find cells, which lead to those deviating ratios, we have to consider all pairwise log-ratios within the observation, which contain the corresponding cell. A crucial issue is to find a good rule of thumb concerning how many pairwise log-ratios have to deviate in order to flag the cell in the original matrix as an outlier.

All mentioned methods were applied on the household expenditure data from Albania. Representing the obtained row- and cell-wise information in a cell map and combining it with the actual data values gives valuable insight into the compositional structure of outlying observations. Not only can we deduce which household's expenditures deviate from the rest, the cellwise information even indicates which particular variables are responsible for this outlying behavior. Sorting the data set by specific variables gives even more insight into the compositional structure and outlying properties of an observation.

Finally, we conducted a simulation study based on the Albanian household expenditure data set to assess the accuracy of our proposed methods. Generally, the algorithms working on the pairwise log-ratio matrix deliver the best results for cell- and row-wise outlier detection, given the underlying simulation design. Algorithms based on an isometric log-ratio transformation struggle to detect cellwise outliers within the original data and often seem to interpret them as rowwise outliers.

Appendix

A Robust imputation methods

In many of our presented algorithms, firstly missing parts of the data are imputed. Among others, the following two *robust imputation methods* for compositional data are available for use and are briefly described here. The detailed description can be found in Hron et al. [2008].

k-Nearest Neighbor Imputation

k-nearest neighbor imputation usually uses the Euclidean distance measure. Since compositional data are represented only in the simplex sample space, a different distance measure has to be used, like the Aitchison distance, being defined for two compositions $\boldsymbol{x} = (x_1, \ldots, x_D)$ and $\boldsymbol{y} = (y_1, \ldots, y_D)$ as

$$d_a(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\frac{1}{D} \sum_{i=1}^{D-1} \sum_{j=i+1}^{D} \left(\ln \frac{x_i}{x_j} - \ln \frac{y_i}{y_j} \right)^2}.$$

Thus, the Aitchison distance considers the property that compositional data include their information only in the ratios between the parts.

Once the k-nearest neighbors to an observation with missing parts have been identified, their information is used to estimate the missings. For reasons of robustness, the estimation is based on using medians rather than means. If the compositional data do not sum up to a constant, which is the case in the investigated data, it is important to use an adjustment according the sum of all parts prior to imputation. For details, see [Hron et al., 2008].

Iterative Model-Based Imputation

Another approach initializes the missing values with the proposed k-nearest neighbor approach. Accordingly the data are transformed to the $D-1$ dimensional real space using the ilr transformation. Let d_e denote the Euclidean distance. The ilr transformation holds the so-called isometric property (see Egozcue and Pawlowsky-Glahn [2005]),

$$d_a(\boldsymbol{x}, \boldsymbol{y}) = d_e(ilr(\boldsymbol{x}), ilr(\boldsymbol{y})).$$

Consequently, one can use standard statistical methods like multiple linear regression, that work correctly in the Euclidean space. In this case the ilr-transformation as in Formula (2.6) is used again. Here, the compositional part $\boldsymbol{x}_1$ includes the highest amount of missings, $\boldsymbol{x}_2$ the next highest, and so on. Thus, when performing a regression of $\boldsymbol{z}_1$ on $\boldsymbol{z}_2, \ldots, \boldsymbol{z}_{D-1}$, only $\boldsymbol{z}_1$ will be influenced by the initialized missings in $\boldsymbol{x}_1$, but not the remaining ilr variables.

The idea of the procedure is thus to iteratively improve the estimation of the missing values. After the regression of $\boldsymbol{z}_1$ on $\boldsymbol{z}_2, \ldots, \boldsymbol{z}_{D-1}$, the results are back-transformed to the simplex, and the cells that were originally missing are updated. Next we consider the variable which originally has the second highest amount of missings, and the same regression procedure is applied in the ilr space. After each variable containing missings has been processed, one can start the whole procedure again until the estimated missings stabilize. The detailed description of this algorithm can be found in Hron et al. [2008]. The authors propose to use robust regression, e.g. LTS regression, especially if outliers might be present in the data.

B Addtional robust estimates used in the *detectDeviatingCells* algorithm

The majority of the algorithm described in Section 3.4 is based on the *detectDeviatingCells* algorithm proposed by Rousseeuw and Van den Bossche [2016]. Within the algorithm they use additional bivariate methods, i.e. methods that operate on two data columns, call them j and h.

They propose a robust correlation estimate by starting from the initial estimate

$$u_{jh} = \frac{((robScale_i(z_{ij} + z_{ih}))^2 - (robScale_i(z_{ij} - z_{ih}))^2}{4} \tag{1}$$

(see Gnanadesikan and Kettenring [1972]) which is capped to lie between -1 and 1 (this assumes that the columns of matrix $\boldsymbol{Z}$ were already normalized and centered at 0). This u_{jh} implies a ellipse around (0,0) with the same coverage probability p as in (3.9). Then $robCorr$ is defined as the plain product-moment correlation of the data points (z_{ij}, z_{ih}) inside the ellipse.

For the slope we again assume the columns were already centered, but they need not be normalized. The authors propose the initial slope estimate

$$b_{jh} = \text{med}_{i=1}^{n} \left(\frac{z_{ij}}{z_{ih}} \right),$$

where fractions with zero denominator are first excluded. For every i the raw residual can be computed using

$$r_{ijh} = z_{ij} - b_{jh} z_{ih}.$$

Finally the plain least squares regression line without intercept is computed on the points for which $|r_{ijh}| \leq c \cdot robScale_i(r_{ijh})$ where c is the constant Equation (3.9). *robSlope* is then defined as the slope of that line.

C R functions

Function `detectOutCell`

```
#' Detect row- and cell-wise outliers in compositional data with structural
#'  zeros
#'
#' @description This function uses robust regression and distance methods to
#'  detect row- and cell-wise outliers in the data. The analysis of the
#'  compositional data is based on working in coordinates (using an isometric
#'  log-ratio transformation) or analysing pairwise log-ratios. To deal with
#'  structural zeros, data is either imputed using robust imputation methods or
#'  the data set is divided into unique subsets (patterns) according to the
#'  zero-pattern occuring in each row.
#'
#' @param x a data.frame or matrix
#' @param add.vars \code{colnames} of additional (grouping-)variables in
#'  \code{x}, which should not be treated as compositional data. These variables
#'  will not be ilr-transformed or be part of pairwise log-ratio calculations,
#'  but will be consulted in the regression analysis of the \code{"imputation"}
#'  and \code{"subset"} approach.
#' @param method method for detecting outliers. Available methods are:
#'  \itemize{
#'    \item \code{"imputation"}
#'    \item \code{"subset"}
#'    \item \code{"pairlog"}
#'    \item \code{"pairlogr"}
#'  }
#' @param imp imputation method. Supported methods are:
#'  \itemize{
#'    \item \code{"kNNA"}
#'    \item \code{"kNN"} from package \code{VIM}
#'    \item \code{"irmi"}
#'    \item \code{"fry"}
#'  }
#' @param reg.m method used for regression analysis. Supported methods are:
#'  \itemize{
#'    \item \code{"ltsReg"} from package \code{robustbase}
#'    \item \code{"lmrob"} from package \code{robustbase}
#'    \item \code{"rlm"} from package \code{MASS}
#'  }
#' @param suppressOutput suppress additional output provided by the algorithms
#'
#' @return An object of class \sQuote{detectOutCell} containing:
#'  \tabular{ll}{
#'    \code{x} \tab the original data.frame \cr
#'    \code{method} \tab method used for detecting outliers \cr
#'    \code{reg.method} \tab the regression method specified in the function
#'      call and used for the regression analysis \cr
#'    \code{reg.vars} \tab variables containing the compositional data \cr
#'    \code{add.vars} \tab additional (grouping-)variables used in the
#'      regression analysis \cr
#'    \code{imputation.approach} \tab a list, containing the results of the
#'     imputation approach, if this approach was specified in parameter
#'     \code{method} \cr
#'   \code{subset.approach} \tab a list, containing the results of the subset
#'     approach, if this approach was specified in parameter \code{method} \cr
#'   \code{pairlog.approach} \tab a list, containing the results of the pairlog
#'     approach, if this approach was specified in parameter \code{method} \cr
#'   \code{pairlogr.approach} \tab a list, containing the results of the
#'     pairlogr approach, if this approach was specified in parameter
#'     \code{method} \cr
#'  }
#'
#' @details For details on the implemented algorithms see Beisteiner & Templ
#'  (2016).
#'
#' @references Beisteiner, L. and Templ, M. (2016). \emph{Exploratory Tools for
#'  Cellwise Outlier Detection in Compositional Data with Structural Zeros}.
#'  Unpublished master's thesis, Vienna University of Technology
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#'
#' @examples
#' data(expenditures)
#' x <- cbind(expenditures, group = rep(c(1, 2)))
#' y <- detectOutCell(x, add.vars = c("group"))
detectOutCell <- function(x, add.vars = NULL, method = c("imputation", "subset",
                                                         "pairlog", "pairlogr"),
                         imp = NULL, reg.m = "lmrob", suppressOutput = FALSE){

  wnq <- function(string, qwrite = 1){ # auxiliary function
    # writes a line without quotes
    if(qwrite == 1) write(noquote(string), file = "", ncolumns = 100)
  }

  x <- as.data.frame(x)
  row.names(x) <- seq(1, nrow(x))

  l.method1 <- NULL
  l.method2 <- NULL
  l.method3 <- NULL
  l.method4 <- NULL

  # check structure of data set beforehand
  # get columns suitable for computing Mahalanobis distances
  if (!all(add.vars %in% colnames(x))){
    stop("Additional variables not part of provided data.")
  }


  method.check <- sapply(method, function(x){
    if (x %in% c("imputation", "Imputation", "Imp", "imp")){
      return("imputation")
    } else if (x %in% c("subset", "Subset", "sub", "Sub")){
      return("subset")
    } else if (x %in% c("pairLog", "pairlog", "pairl", "pairL")){
      return("pairlog")
    } else if (x %in% c("pairLogR", "pairlogr", "pairLogr", "pairlogR",
                        "pairlr", "pairLR")){
      return("pairlogr")
    } else {
      stop("Wrong method for parameter method specified")
    }
  } , USE.NAMES = FALSE)

  method <- method.check

  if (imp %in% c("knn", "KNN", "kNN") || is.null(imp)){
    imp <- "kNN"
  } else if (imp %in% c("impKNNa","knna","KNNa","kNNa")){
    imp <- "KNNa"
  } else if (imp %in% c("IRMI","irmi","Irmi")){
    imp <- "irmi"
  } else if (impute %in% c("fry", "Fry", "FRY")){
    imp <- "fry"
  } else {
    stop("Wrong method for imputation specified")
  }

  if (reg.m %in% c("ltsReg", "ltsreg")){
    reg.m <- "ltsReg"
  } else if (reg.m %in% c("lmrob", "lmRob")) {
    reg.m <- "lmrob"
  } else if (reg.m %in% c("rlm", "Rlm")) {
    reg.m <- "rlm"
  } else {
    stop("Wrong method for regression specified")
  }

  p <- ncol(x)
  ind.excl <- which(colnames(x) %in% c(add.vars))
  if (length(ind.excl) == 0){
    ind <- (1:p)
    ind.excl <- NULL
  } else {
    ind <- (1:p)[-ind.excl]
  }

  reg.vars <- colnames(x)[ind]

  colMHD <- checkDataFrame(x, reg.vars, suppressOutput = suppressOutput)
  y <- x[, ind]

  if (any(is.na(y)) & any(y == 0, na.rm = TRUE)){
    warning("The data includes NA's and zeros. \n
            Impute the missing values first, otherwise they are treated as
            zeros")
  }

```

```r
# set 0 to NA
y[y == 0] <- NA

# get rows that will be imputed
y.imp <- apply(is.na(y), 1, any)

if (any(method %in% c("imputation", "pairlog", "pairlogr"))){
  # 1) Impute
  if (imp == "KNNa"){
    yi <- impKNNa(y)$xImp
  } else if (imp == "kNN"){
    yi <- kNN(y, imp_var = FALSE)
  } else if (imp == "fry"){
    yi <- rmzero(y, minval=0.01, delta=0.01)
  } else if (imp == "irmi"){
    yi <- impCoda(y, init="geometricmean")$xImp
  } else {
    stop("Wrong method for imputation specified")
  }
}

if ("imputation" %in% method){
  if (suppressOutput == FALSE){
    wnq(paste0("+------------------------------------------------------------",
               "------------------+"))
    wnq(paste0("| Performing imputation approach                             ",
               "                  |"))
    wnq(paste0("+------------------------------------------------------------",
               "------------------+\n"))
  }
  # 2) Imputation approach
  # check if all reg.vars are valid for MHD computation (this should always
  # be true, otherwise even ilr-transformation would not make sense)
  if(all(colMHD[ind])){
    # vector, indicating which variables are valid for MHD computation
    # FALSE, rep(.,.) ... compositional variables are always used, except for
    #                     the first one (we are only computing MHDs in the X-
    #                     space, z_1 = y-space)
    ind.mhd <- c(FALSE, rep(TRUE, length(ind)-2), colMHD[ind.excl])
    cols.new.ilr <- c(paste0("z_", seq(1:(length(ind)-1))))
    names(ind.mhd) <- c(cols.new.ilr, add.vars)

    # number of valid variables
    sum.mhd <- sum(ind.mhd)
  } else {
    stop(paste("The provided compositional variables used for regression ",
               " are not applicable.", sep = ""))
  }

  for (i in ind){
    ind.new <- c(i, ind[-i])
    z_ilr <- as.data.frame(cbind(isomLR(yi[, ind.new]), x[, add.vars]))
    colnames(z_ilr) <- c(cols.new.ilr, add.vars)

    if (reg.m == "ltsReg"){
      tmp.1 <- ltsReg(z_1 ~ ., data = z_ilr)
      r.ilr <- tmp.1$residuals / tmp.1$scale
      out.ilr <- as.integer(abs(r.ilr) > 2.5)

      # compute robust MHD
      tmp.2 <- covMcd(z_ilr[, ind.mhd])
      mhd <- mahalanobis(z_ilr[, ind.mhd], tmp.2$center, tmp.2$cov)
      out.ilr.x <- as.integer(mhd > qchisq(0.975, sum.mhd))
    } else if (reg.m == "lmrob") {
      ## ilr-regression
      tmp.1 <- lmrob(z_1 ~ ., data = z_ilr)
      r.ilr <- tmp.1$residuals / tmp.1$scale
      out.ilr <- as.integer(abs(r.ilr) > 2.5)

      # compute robust MHD
      tmp.2 <- covMcd(z_ilr[, ind.mhd])
      mhd <- mahalanobis(z_ilr[, ind.mhd], tmp.2$center, tmp.2$cov)
      out.ilr.x <- as.integer(mhd > qchisq(0.975, sum.mhd))
    } else if (reg.m == "rlm"){
      tmp.1 <- rlm(z_1 ~ ., data = z_ilr)
      r.ilr <- tmp.1$residuals / tmp.1$s
      out.ilr <- as.integer(abs(r.ilr) > 2.5)

      # compute robust MHD
      tmp.2 <- covMcd(z_ilr[, ind.mhd])
      mhd <- mahalanobis(z_ilr[, ind.mhd], tmp.2$center, tmp.2$cov)
      out.ilr.x <- as.integer(mhd > qchisq(0.975, sum.mhd))
    }

    l.method1[[i]] <- list(trafo.colorder = ind.new, out.ilr = out.ilr,
                           out.ilr.x = out.ilr.x)
  }

  mat.out.ilr <- matrix(unlist((lapply(l.method1, function(x) x$out.ilr))),
                        nrow = nrow(x))
  mat.out.ilr <- as.data.frame(mat.out.ilr)
  colnames(mat.out.ilr) <- paste0(reg.vars, "_out.ilr")
  mat.out.ilr.x <- matrix(unlist((lapply(l.method1,
                                   function(x) x$out.ilr.x))),
                          nrow = nrow(x))
  mat.out.ilr.x <- as.data.frame(mat.out.ilr.x)
  colnames(mat.out.ilr.x) <- paste0(reg.vars, "_out.ilr.x")

  mat.cell <- matrix(0, ncol = ncol(mat.out.ilr), nrow = nrow(mat.out.ilr))
  out.mult <- which(rowSums(mat.out.ilr) >= floor(ncol(mat.out.ilr)/2))
  out.mult <- c(out.mult,
                which(rowSums(mat.out.ilr.x) == ncol(mat.out.ilr.x)))
  out.mult <- unique(out.mult)
  out.cell <- which(mat.out.ilr == 1 & mat.out.ilr.x == 0)
  mat.cell[out.mult, ] <- 3
  mat.cell[out.cell] <- 1
  mat.cell[is.na(y)] <- 4

  mat.cell <- as.data.frame(mat.cell)
  colnames(mat.cell) <- reg.vars

  l.method1$mat.out.ilr <- mat.out.ilr
  l.method1$mat.out.ilr.x <- mat.out.ilr.x
  l.method1$mat.cell <- mat.cell
}

if ("subset" %in% method){
  if (suppressOutput == FALSE){
    wnq(paste0("+----------------------------------------------------------",
               "-------------------+"))
    wnq(paste0("| Performing subset approach                             ",
               "                 |"))
    wnq(paste0("+----------------------------------------------------------",
               "-------------------+\n"))
  }
  # 4) subset-approach without imputation
  w <- is.na(y)
  w <- apply(w, 2, as.integer)
  s <- apply(w, 1, paste, collapse = "")
  ys <- cbind(y, x[, add.vars])
  colnames(ys) <- c(colnames(y), add.vars)
  ys <- split(ys, s)
  names(ys) <- gsub("0", "x", names(ys))
  names(ys) <- gsub("1", "0", names(ys))

  # if one group is too small report an error
  check <- as.numeric(unlist(lapply(ys, nrow)))
  w <- which(check < 2*(ncol(x)-1) + 2)
  if(length(w) > 0 && suppressOutput == FALSE){
    m <- missPatterns(y)$tabcombPlus
    cat("\n The following subsets:\n")
    print(m[w, ]*1)
    cat("\n are too small for evaluation in each subcomposition!")
    cat("\n Subsets must be larger than 2*ncol(x) + 1.")
    cat("Therefore possible outliers in these subsets can not be detected!")
  }

  # exclude NA columns
  ys <- lapply(ys, function(x){
    x <- x[, !is.na(x[1, ]), drop = FALSE]
    x
  })

  j <- 1
  num.reg <- 0

  mat.out.ilr <- matrix(NA, ncol = ncol(y), nrow = nrow(y))
  mat.out.ilr.x <- matrix(NA, ncol = ncol(y), nrow = nrow(y))

  for (yj in ys){
    # check if yj is big enough, w...groups, which are too small
    # if big enough, proceed normally
    if (!(j %in% w)){
      p <- ncol(yj)
      ind.excl <- which(colnames(yj) %in% c(add.vars))

      if (length(ind.excl) == 0){
        ind <- (1:p)
        ind.excl <- NULL
        ind.orig <- which(colnames(x) %in% colnames(yj))
      } else {
        ind <- (1:p)[-ind.excl]
        ind.orig <- which(colnames(x) %in% colnames(yj)[-ind.excl])
      }

      if (suppressOutput == FALSE){
        wnq(paste("\n Checking variables for RD-calculation of subset ",
                  names(ys)[j], ".\n", sep = ""))
      }
      colMHD <- checkDataFrame(yj, colnames(yj)[ind],
                               suppressOutput = suppressOutput)
      # check if all reg.vars are valid for MHD computation (this should
      # always be true, otherwise even ilr-transformation would not make
      # sense)
      if(all(colMHD[ind])){
        # vector, indicating which variables are valid for MHD computation
        # FALSE, rep(.,.) ... compositional variables are always used, except
        #                     for the first one (we are only computing MHDs in
        #                     the X-space, z_1 = y-space)
        ind.mhd <- c(FALSE, rep(TRUE, length(ind)-2), colMHD[ind.excl])
        cols.new.ilr <- c(paste0("z_", seq(1:(length(ind)-1))))
        names(ind.mhd) <- c(cols.new.ilr, add.vars)

        # number of valid variables
        sum.mhd <- sum(ind.mhd)
      } else {
        stop(paste("The provided compositional variables used for ",
                   " regression are not applicable.", sep = ""))
      }

      num.reg <- num.reg + length(ind)

      row.num <- as.integer(rownames(yj))
      out.ilr <- rep(NA, nrow(y))
      out.ilr.x <- rep(NA, nrow(y))

      l.tmp <- list()
      for (i in ind){
        ind.new <- c(i, ind[-i])
        z_ilr <- as.data.frame(cbind(isomLR(yj[, ind.new]), yj[, ind.excl]))
        colnames(z_ilr) <- c(cols.new.ilr, add.vars)

        if (reg.m == "ltsReg"){
          tmp.1 <- ltsReg(z_1 ~ ., data = z_ilr)
          r.ilr <- tmp.1$residuals / tmp.1$scale
          out.ilr[row.num] <- as.integer(abs(r.ilr) > 2.5)

          # compute robust distances
          tmp.2 <- sign2(z_ilr[, ind.mhd])
          out.ilr.x[row.num] <- (1-tmp.2$wfinal01)

        } else if (reg.m == "lmrob") {
          tmp.1 <- lmrob(z_1 ~ ., data = z_ilr)
          r.ilr <- tmp.1$residuals / tmp.1$scale
          out.ilr[row.num] <- as.integer(abs(r.ilr) > 2.5)

          # compute robust distances
          tmp.2 <- sign2(z_ilr[, ind.mhd])
          out.ilr.x[row.num] <- (1-tmp.2$wfinal01)

        } else if (reg.m == "rlm"){
          tmp.1 <- rlm(z_1 ~ ., data = z_ilr)
          r.ilr <- tmp.1$residuals / tmp.1$s
          out.ilr[row.num] <- as.integer(abs(r.ilr) > 2.5)

          # compute robust MHD
          tmp.2 <- sign2(z_ilr[, ind.mhd])
          out.ilr.x[row.num] <- (1-tmp.2$wfinal01)
        }

        mat.out.ilr[row.num, ind.orig[i]] <- out.ilr[row.num]
        mat.out.ilr.x[row.num, ind.orig[i]] <- out.ilr.x[row.num]

        l.tmp[[i]] <- list(orig.cols = ind.orig, trafo.colorder = ind.new,
                           out.ilr = out.ilr, out.ilr.x = out.ilr.x)
      }
      # if group too small, outliers can not be detected
      # therefore insert 0 in vector and NULL in trafo.colorder,
      # due to no regression takes place
    } else {
      ind.excl <- which(colnames(yj) %in% c(add.vars))

      if (length(ind.excl) == 0){
        ind.orig <- which(colnames(x) %in% colnames(yj))
      } else {
        ind.orig <- which(colnames(x) %in% colnames(yj)[-ind.excl])
      }

      num.reg <- num.reg + 1

      row.num <- as.integer(rownames(yj))
      out.ilr <- rep(NA, nrow(y))
      out.ilr.x <- rep(NA, nrow(y))
      out.ilr[row.num] <- 0
      out.ilr.x[row.num] <- 0

      mat.out.ilr[row.num, ind.orig] <- 0
      mat.out.ilr.x[row.num, ind.orig] <- 0

      l.tmp <- list()
      l.tmp <- list(orig.cols = ind.orig, trafo.colorder = NULL,
                    out.ilr = out.ilr, out.ilr.x = out.ilr.x)
    }
    l.method2[[names(ys)[j]]] <- l.tmp
    j <- j + 1
  }

  mat.out.ilr <- as.data.frame(mat.out.ilr)
  colnames(mat.out.ilr) <- paste0(reg.vars, "_out.ilr")
  mat.out.ilr.x <- as.data.frame(mat.out.ilr.x)
  colnames(mat.out.ilr.x) <- paste0(reg.vars, "_out.ilr.x")

  mat.cell <- matrix(0, ncol = ncol(mat.out.ilr), nrow = nrow(mat.out.ilr))
  out.mult <- which(rowSums(mat.out.ilr, na.rm = TRUE) >=
                      floor(ncol(mat.out.ilr)/2))
  out.mult <- c(out.mult,
                which(rowSums(mat.out.ilr.x, na.rm = TRUE) ==
                        (ncol(mat.out.ilr.x) - rowSums(is.na(y)))))
  out.mult <- unique(out.mult)
  out.cell <- which(mat.out.ilr == 1 & mat.out.ilr.x == 0)
  mat.cell[out.mult, ] <- 3
  mat.cell[out.cell] <- 1
  mat.cell[is.na(y)] <- 4

  mat.cell <- as.data.frame(mat.cell)
  colnames(mat.cell) <- reg.vars

  l.method2$mat.out.ilr <- mat.out.ilr
  l.method2$mat.out.ilr.x <- mat.out.ilr.x
  l.method2$mat.cell <- mat.cell
}

if ("pairlog" %in% method){
  if (suppressOutput == FALSE){
    wnq(paste0("+------------------------------------------------------------",
               "------------------+"))
    wnq(paste0("| Performing pairwise log-ratio approach                    ",
               "                 |"))
    wnq(paste0("+------------------------------------------------------------",
               "------------------+\n"))
  }
  tmp <- detectDeviatingLogs(yi, suppressOutput = suppressOutput)
  mat.cell <- tmp$mat.cell
  mat.cell[is.na(y)] <- 4

  mat.cell <- as.data.frame(mat.cell)
  colnames(mat.cell) <- reg.vars

  l.method3$u <- tmp$u
  l.method3$mat.pair <- tmp$mat.pair
  l.method3$rd.pair <- tmp$rd.pair
  l.method3$mat.cell <- mat.cell
}

```

```r
  if ("pairlogr" %in% method){
    if (suppressOutput == FALSE){
      wnq(paste0("+------------------------------------------------------------",
                 "------------------+"))
      wnq(paste0("| Performing pairwise log-ratio Rousseeuw approach   ",
                 "                 |"))
      wnq(paste0("+------------------------------------------------------------",
                 "------------------+\n"))
    }
    tmp <- detectDeviatingLogsRou(yi, suppressOutput = suppressOutput)
    mat.cell <- tmp$mat.cell
    mat.cell[is.na(y)] <- 4

    mat.cell <- as.data.frame(mat.cell)
    colnames(mat.cell) <- reg.vars

    l.method4$x.log.cell <- tmp$x.log.cell
    l.method4$mat.cell <- mat.cell
  }

  rlist <- list(x = x, method = method, reg.method = reg.m, reg.vars = reg.vars,
               add.vars = add.vars)
  rlist[["imputation.approach"]] <- l.method1
  rlist[["subset.approach"]] <- l.method2
  rlist[["pairlog.approach"]] <- l.method3
  rlist[["pairlogr.approach"]] <- l.method4
  class(rlist) <- "detectOutCell"
  return(rlist)
}
```

Function detectDeviatingLogs

```r
#' Pairwise log-ratio approach (workhorse detectOutCell)
#'
#' @description Workhorse function for the \code{"pairlog"} approach in
#'  \code{detectOutCell}, detecting row- and cellwise outliers using the
#'  pairwise log-ratio matrix between the compositional variables.
#'
#' @param x a \code{matrix}
#' @param suppressOutput suppress additional output provided by the algorithm
#'
#' @return A list containing:
#'  \tabular{ll}{
#'    \code{u} \tab a 0/1-matrix, resulting from the initial univariate outlier
#'      detection of function \code{detectUnivOut}. 1 = univariate outlier,
#'      0 = regular data cell \cr
#'    \code{mat.pair} \tab results of the regression analysis between good and
#'      bad variables/ratios \cr
#'    \code{rd.pair} \tab results of the distance analysis between good and bad
#'      variables/ratios \cr
#'    \code{mat.cell} \tab row- and cellwise outlier information \cr
#'  }
```

```r
#'
#'
#' @details Entries in \code{mat.cell} represent:
#'  \itemize{
#'    \item 0 - regular cells
#'    \item 1 - cellwise outliers
#'    \item 2 - bivariate cellwise outliers
#'    \item 3 - rowwise outliers
#'    \item 4 - missing values
#'  }
#'  For details on the algorithm see Beisteiner & Templ (2016).
#'
#' @references Beisteiner, L. and Templ, M. (2016). \emph{Exploratory Tools for
#'  Cellwise Outlier Detection in Compositional Data with Structural Zeros}.
#'  Unpublished master's thesis, Vienna University of Technology
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectOutCell}}
detectDeviatingLogs <- function(x, suppressOutput = FALSE){

  wnq <- function(string, qwrite = 1){ # auxiliary function
    # writes a line without quotes
    if(qwrite == 1) write(noquote(string), file = "", ncolumns = 100)
  }

  x <- as.data.frame(x)

  x.cell <- matrix(0, nrow = nrow(x), ncol = ncol(x))
  mat.pair.sum <- NULL
  rd.pair.sum <- NULL

  u <- detectUnivOut(x)
  s <- apply(u, 1, paste, collapse = "")
  xs <- split(x, s)

  cols <- 1:ncol(x)
  group.names <- names(xs)

  for (i in 1:length(xs)){
    if (suppressOutput == FALSE){
      wnq(paste("Processing group ", group.names[i], " (", i, "/", length(xs),
                ")", sep =""))
    }
    s
    y <- xs[[i]]
    cols.bad <- which(strsplit(group.names[i], "")[[1]] == "1")
    l.bad <- length(cols.bad)

    if (l.bad == 0){
      cols.good <- cols
      cols.bad <- NULL
    } else {
      cols.good <- cols[-cols.bad]
    }

    ind.obs <- as.integer(rownames(y))

    l.good <- length(cols.good)
    l.ind <- length(ind.obs)

    tmp <- pairwLogRatios(x, cols.good, cols.bad)
    Z <- tmp$Z

    # A) compute regression models
    mat.pair <- data.frame()
    if (l.good > 1){
      for (model in tmp$reg.models){
        tmp.1 <- lmrob(formula(model), data = Z)
        r <- tmp.1$residuals / tmp.1$scale
        out <- as.integer(abs(r) > 2.5)

        # compute robust MHD
        tmp.2 <- sign2(Z[, tmp$x.vars, drop = FALSE])
        mat.pair <- rbind(mat.pair,
                          matrix(c(ind.obs, out[ind.obs],
                                   1-tmp.2$wfinal01[ind.obs]),
                                 ncol = 3))
      }
    }

    rd.pair <- data.frame()
    n.rat <- c()

    # B) compute RDs
    # -- between good ratios
    if (l.good > 1){
      tmp.2 <- sign2(Z[, tmp$rd.good, drop = FALSE])
      rd.pair <- rbind(rd.pair, matrix(c(ind.obs, 1-tmp.2$wfinal01[ind.obs]),
                                       ncol = 2))
      n.rat <- c(n.rat, "g")
    }

    # -- between bad ratios
    if (l.bad > 1){
      tmp.2 <- sign2(Z[, tmp$rd.bad, drop = FALSE])
      rd.pair <- rbind(rd.pair, matrix(c(ind.obs, 1-tmp.2$wfinal01[ind.obs]),
                                        ncol = 2))
      n.rat <- c(n.rat, "b")
    }

    rd.pair <- cbind(rep(n.rat, each = l.ind), rd.pair)
    colnames(rd.pair) <- c("rat", "row_num", "rd")

    # c) interpret results
    for (j in n.rat){
      for (k in ind.obs){
        mat.sub <- subset(rd.pair, (rat == j) & (row_num == k))
        if (mat.sub[, "rd"] == 1){
          if (j == "g"){
            x.cell[k, cols.good] <- 3
          } else if (j == "b"){
            x.cell[k, cols.bad] <- 3
          }
        }
      }
    }

    if (l.bad > 0 & l.good > 1){
      x_var <- c(rep(cols.bad, each = (l.ind * l.good)))
      if (l.bad > 1){
        x_var <- c(x_var, rep(0, (l.ind * (l.bad-1)*l.bad/2)))
      }

      mat.pair <- cbind(x_var, mat.pair)
      colnames(mat.pair) <- c("x_var", "row_num", "res", "rd")

      for (j in cols.bad){
        for (k in ind.obs){
          mat.sub <- subset(mat.pair, (x_var == j) & (row_num == k))
          if (all(mat.sub[, "res"] == 1) &&
             all(mat.sub[, "rd"] == 0)){
            x.cell[k, j] <- 1
          }
        }
      }

      if (l.bad > 1){
        for (k in ind.obs){
          mat.sub <- subset(mat.pair, (x_var == 0) & (row_num == k))
          j <- which(mat.sub[,"res"] == 1 & mat.sub[,"rd"] == 0)
          y.names <- tail(tmp$y.vars, n = (l.bad-1)*l.bad/2)[j]
          y.names <- substring(y.names, 6)
          y.names <- c(as.integer(unlist(strsplit(y.names, "_x"))))
          y.names <- unique(y.names)
          x.cell[k, y.names] <- 2
        }
      }
    }

    mat.pair.sum <- rbind(mat.pair.sum, mat.pair)
    rd.pair.sum <- rbind(rd.pair.sum, rd.pair)
   }

   return(list(u = u, mat.pair = mat.pair.sum, rd.pair = rd.pair.sum,
               mat.cell = x.cell))
  }
```

Function `detectDeviatingLogsRou`

```r
#' Detect deviating cells on pairwise log-ratios (workhorse detectOutCell)
#'
#' @description Workhorse function for the \code{"pairlogr"} approach in
#'  \code{detectOutCell}, detecting row- and cellwise outliers using the
#'  \emph{detectDeviatingCells} method proposed by Rousseeuw & Van den Bossche
#'  (2016).
#'
#' @param x a \code{matrix}
#' @param suppressOutput suppress additional output provided by the algorithm
#'
#' @return A list containing:
#'  \tabular{ll}{
#'    \code{mat.cell} \tab row- and cellwise outlier information \cr
#'    \code{x.log.cell} \tab row- and cellwise outlier information of the
#'      pairwise log-ratio matrix
#'  }
#'
#'
#' @details Entries in \code{mat.cell} represent:
#'  \itemize{
#'    \item 0 - regular cells
#'    \item 1 - cellwise outliers
#'    \item 2 - bivariate cellwise outliers
#'    \item 3 - rowwise outliers
#'    \item 4 - missing values
#'  }
#'  For details on the algorithm see Beisteiner & Templ (2016) and Rousseeuw &
#'  Van den Bossche (2016).
#'
#' @references Beisteiner, L. and Templ, M. (2016). \emph{Exploratory Tools for
#'  Cellwise Outlier Detection in Compositional Data with Structural Zeros}.
#'  Unpublished master's thesis, Vienna University of Technology
#' @references Rousseeuw, P.J. and Van den Bossche, W. (2016). \emph{Detecting
#'  anomalous data cells.} ArXiv e-prints
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectOutCell}}
detectDeviatingLogsRou <- function(x, suppressOutput = FALSE){

  x <- as.data.frame(x)

  ## Rousseeuw
  px <- ncol(x)
  nx <- nrow(x)

  tmp <- pairwLogRatios(x, 1:px, NULL)
  ddc <- DetectDeviatingCells(tmp$X, suppressOutput = suppressOutput)

  p <- ncol(tmp$X)
  n <- nrow(tmp$X)

  if (length(ddc$colInAnalysis) == p && length(ddc$rowInAnalysis) == n){
    x.log.cell <- matrix(0, ncol = p, nrow = n)
    x.log.cell[ddc$indcells] <- 1

    mat.cell <- matrix(0, ncol = px, nrow = nx)
    # rowwise outlier in pair-wise log-ratio matrix are rowwise outlier in x
    mat.cell[ddc$indrows, ] <- 3

    # calculate log-ratio groups
    l.group <- list()
    for (i in 1:(px-1)){
      if (i == 1){
        l.begin <- 1
        l.end <- px - 1
        l.group[[i]] <- c(l.begin:l.end)
      } else {
        l.begin <- sum(px-c(1:(i-1)))+1
        l.end <- sum(px-c(1:i))
        l.group[[i]] <- c(l.begin:l.end)
      }
    }

    for (i in 1:px){
      ind.group <- sapply(1:i, function(j){
        if (j == i){
          if (j != px){
            ind.group <- l.group[[j]]
          }
        } else {
          ind.group <- l.group[[j]][i-j]
        }
      })

      ind.group <- unlist(ind.group)
      out.cell <- rowSums(x.log.cell[, ind.group]) >= floor(px/2)
      mat.cell[out.cell, i] <- 1

      #out.mult <- which(rowSums(mat.cell) == ncol(mat.cell))
      out.mult <- which(rowSums(mat.cell) >= floor(px/2))
      mat.cell[out.mult, ] <- 3
      mat.cell <- data.frame(mat.cell)
      x.log.cell <- data.frame(x.log.cell)
      colnames(mat.cell) <- colnames(x)
      colnames(x.log.cell) <- colnames(tmp$X)
    }
  } else {
    stop("Only ", length(ddc$colInAnalysis), "/", p, " columns and ",
         length(ddc$rowInAnalysis), "/", n, " rows of the pairwise log-ratio ",
         "matrix were analyzed by the algorithm. Results can not be ",
         "interpreted.")
  }

  return(list(mat.cell = mat.cell, x.log.cell = x.log.cell))
}
```

Function print.detectOutCell

```r
#' Print method for objects of class detectOutCell
#'
#' The \code{print}-method summarises the results of function
#'  \code{detectOutCell} by approach and variable.
#'
#' @details The function prints a well-aranged list of the number of cell- and
#'  rowwise outliers detect by each approach and variable.
#'
#' @param x an object of class \sQuote{detectOutCell}
#' @method print detectOutCell
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectOutCell}}
#'
#' @examples
#' data(expenditures)
#' x <- cbind(expenditures, group = rep(c(1, 2)))
#' y <- detectOutCell(x, add.vars = c("group"))
#' print(y)
print.detectOutCell <- function(x){
  cat("Number of cellwise outliers detected by approach and variable:\n\n")

  m.cell <- matrix(nrow = 0, ncol = length(x$reg.vars) + 1)
  v.row <- c()

  for (m in x$method){
    app <- paste0(m, ".approach")
    mat.cell <- x[[app]]$mat.cell
    res.row <- which(rowSums(mat.cell == 3) > 1)
    res.cell <- colSums(mat.cell == 1 | mat.cell == 2)
    res.cell <- c(res.cell, sum(res.cell))

    m.cell <- rbind(m.cell, res.cell)
    v.row <- c(v.row, length(res.row))
  }
  rownames(m.cell) <- x$method
  colnames(m.cell) <- c(x$reg.vars, "sum")

  names(v.row) <- x$method

  print(m.cell)
  cat("\n")
  cat("Number of rowwise outliers detected by approach:\n\n")
  print(v.row)
  cat("\n")
}
```

Function plot.detectOutCell

```
#' Plot cell map for objects of class detectOutCell
#'
#' @description Plot a color coded cell- or block-map of the results of function
#'  \code{detectOutCell}
#'
#' @param x an object of class \code{detectOutCell}
#' @param blocksize if 1 (default) plot results for each cell of the matrix, if
#'  > 1 group neighboring cells in blocks by dimension \code{blockdim}
#' @param blockdim if \code{"row"} always group \code{blocksize} rows together,
#'  if \code{"both"} group the \code{blocksize} neighboring cells by row and
#'  column together
#' @param rownums only plot the specified row numbers, default \code{NULL} plots
#'  all rows
#' @param orderby order data set by variable, either the column name or the
#'  index of the variable within the input data set may be specified; if
#'  \code{NULL} (default) no ordering is done
#' @param decreasing if \code{orderby} is specified, order the variable
#'  ascending (\code{FALSE}) or descending (\code{TRUE})
#' @param labelsx labels for the x-axis, if \code{NULL} use \code{rownames}
#' @param labelsy labels for the y-axis, if \code{NULL} use \code{colnames}
#' @param xtitle title for the x-axis
#' @param ytitle title for the y-axis
#' @param anglex angle of the labels on the x-axis
#' @param sizexy size of title for x-axis and y-axis
#' @param hjustXlabels adjust x-labels: 0 = left, 0.5 = centered, 1 = right
#' @param hjustYlabels adjust y-labels
#' @param base_size adjust base size of x- and y-axis labels
#'
#' @details Cells are visualized by color-coded squares according to the entries
#'  in the result matrix:
#'  \itemize{
#'    \item yellow (value 0) - regular cells
#'    \item red (value 1) - cellwise outliers
#'    \item blue (value 2) - bivariate cellwise outliers
#'    \item black (value 3) - rowwise outliers
#'    \item white (value 4) - missing values
#'  }
#'
#' @method plot detectOutCell
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectOutCell}}
#'
#' @examples
#' data(expenditures)
#' x <- cbind(expenditures, group = rep(c(1, 2)))
#' y <- detectOutCell(x, add.vars = c("group"))
#' plot(y)
#'
#' plot(y, blocksize = 2)
#'
```

```r
#' plot(y, blocksize = 2, blockdim = "both")
plot.detectOutCell <- function(x, blocksize = 1, blockdim = "row",
                              rownums = NULL, orderby = NULL,
                              decreasing = FALSE, labelsx = NULL,
                              labelsy = NULL, xtitle = "", ytitle = "",
                              anglex = 90, sizexy = 1.8, hjustXlabels = 1,
                              hjustYlabels = 1, base_size = 5){

  if (blocksize < 1){
    stop("Invalid blocksize")
  }

  if (!(blockdim %in% c("row", "both")) && blocksize > 1){
    warning(paste0('Can not reduce cell map to dimension "', blockdim, '"!',
                   ' Using default dimension "row" instead.'))
    blockdim <- "row"
  }

  if (is.null(rownums)){
    rownums <- 1:nrow(x$x)
  }

  if (!is.null(orderby)){
    if ((is.numeric(orderby) &&
         between(orderby, 0, ncol(x$x[, x$reg.vars]) + 1, incbounds = FALSE)) |
        orderby %in% x$reg.vars){
      tmp.order <- order(x$x[, orderby], decreasing = decreasing,
                         na.last = decreasing)

      if (is.null(labelsy)){
        labelsy <- rownames(x$x)[tmp.order]
        labesly <- labelsy[rownums]
      }
    }
  }

  if (is.null(labelsx)){
    labelsx <- colnames(x$x[, x$reg.vars])
  }

  if (is.null(labelsy)){
    labelsy <- rownames(x$x)[rownums]
  }

  methods <- paste0(x$method, ".approach")
  gplot <- list()

  for (m in x$method){
    if (blocksize == 1){
      app <- paste0(m, ".approach")
      strTitle <- paste0(m, " app.")
      mat.cell <- x[[app]]$mat.cell

      if (!is.null(orderby)){
```

```
      mat.cell <- mat.cell[tmp.order, ]
    }

    gplot[[m]] <- plotCellMap(mat.cell[rownums, ], labelsx = labelsx,
                              labelsy = labelsy, strTitle = strTitle,
                              anglex = anglex, xtitle = xtitle,
                              ytitle = ytitle, sizexy = sizexy,
                              hjustXlabels = hjustXlabels,
                              hjustYlabels = hjustYlabels,
                              base_size = base_size)

  } else {
    app <- paste0(m, ".approach")
    strTitle <- paste0(m, " app.")
    mat.cell <- x[[app]]$mat.cell

    if (!is.null(orderby)){
      mat.cell <- mat.cell[tmp.order, ]
    }

    gplot[[m]] <- plotBlockMap(mat.cell[rownums, ],
                               labelsx = labelsx, labelsy = labelsy,
                               strTitle = strTitle, blocksize = blocksize,
                               blockdim = blockdim, anglex = anglex,
                               xtitle = xtitle, ytitle = ytitle,
                               sizexy = sizexy, autolabel = TRUE,
                               base_size = base_size)
  }
 }
 multiplot(plotlist = gplot, cols = length(gplot))
}
```

Function plotCellMap

```
#' Plot cell map (workhorse plot.detectOutCell)
#'
#' @description Workhorse function for \code{plot.detectOutCell}, plotting a
#'  color coded cell map
#'
#' @param x a \code{matrix}
#' @param labelsx labels for the x-axis, if \code{NULL} use \code{rownames}
#' @param labelsy labels for the y-axis, if \code{NULL} use \code{colnames}
#' @param strTitle title of the cellMap
#' @param anglex angle of the labels on the x-axis
#' @param xtitle title for the x-axis
#' @param ytitle title for the y-axis
#' @param sizexy size of title for x-axis and y-axis
#' @param hjustXlabels adjust x-labels: 0 = left, 0.5 = centered, 1 = right
#' @param hjustYlabels adjust y-labels
#' @param base_size adjust base size of x- and y-axis labels
#'
#' @return ggp An object of class \sQuote{ggplot} containing the plotted cell
```

```r
#'  map
#'
#' @details Cells are visualized by color-coded squares according to the entries
#'  in the result matrix:
#'  \itemize{
#'    \item yellow (value 0) - regular cells
#'    \item red (value 1) - cellwise outliers
#'    \item blue (value 2) - bivariate cellwise outliers
#'    \item black (value 3) - rowwise outliers
#'    \item white (value 4) - missing values
#'  }
#'  This method is an adapted version of the plot method proposed by Rousseeuw &
#'  Van den Bossche (2016).
#'
#' @references Rousseeuw, P.J. and Van den Bossche, W. (2016). \emph{Detecting
#'  anomalous data cells.} ArXiv e-prints
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @seealso \code{\link{plot.detectOutCell}}
#' @note License: GPL-2
plotCellMap <- function(x, labelsx, labelsy, strTitle, anglex = 90,
                        xtitle = "", ytitle = "", sizexy = 1.8,
                        hjustXlabels = 1, hjustYlabels = 1, base_size = 10){

  labelsy <- rev(labelsy) # will be plotted from bottom to top
  n <- nrow(x)
  p <- ncol(x)

  # Melt data matrices for cellMap
  Xdf <- data.frame(cbind(seq(1, n, 1), x))
  colnames(Xdf) <- c("row_num", colnames(x))
  rownames(Xdf) <- NULL
  Xdf$row_num <- with(Xdf, reorder(row_num, seq(n, 1, -1)))
  mX <- melt(Xdf, id.var = "row_num", value.name = "CatNr")

  Ddf <- data.frame(cbind(seq(1, n, 1), x))
  colnames(Ddf) <- c("row_num", colnames(x))
  rownames(Ddf) <- NULL
  Ddf$row_num = with(Ddf, reorder(row_num, seq(n, 1, -1)))
  mD <- melt(Ddf, id.var = "row_num")

  # Combine melted data
  mX$data <- mD$value
  mX$rescaleoffset <- 100 * mX$CatNr
  gradientends <- rep(c(0, 100, 200, 300, 400), each = 2)

  colorends <- c("khaki1", "khaki1", "red", "red", "blue", "blue",
                 "black", "black", "white", "white")

  ggp = ggplot(data = mX, aes(variable, row_num)) +
    geom_tile(aes(fill = rescale(rescaleoffset, from = c(0, 400))),
              colour = "white") +
    scale_fill_gradientn(colours = colorends,
                         values = rescale(gradientends),
                         rescaler = function(x, ...) x, oob = identity) +
    ggtitle(strTitle) +
    coord_fixed() +
    theme_classic(base_size = base_size*1) +
    labs(x = xtitle, y = ytitle) +
    scale_x_discrete(expand = c(0, 0), labels = labelsx) +
    scale_y_discrete(expand = c(0, 0), labels = labelsy) +
    theme(legend.position = "none", axis.ticks = element_blank(),
          plot.title = element_text(size = base_size*2, vjust = 1,
                                    face = "bold"),
          # hjust = 1 right-adjusts the labels on the x-axis:
          axis.text.x = element_text(size = base_size*1.8, angle = anglex,
                                   hjust = hjustXlabels, vjust = 0.5,
                                   colour = "black"),
          axis.text.y = element_text(size = base_size*1.8, angle = 0,
                                   hjust = hjustYlabels, colour = "black"),
          axis.title.x = element_text(colour = "black", size = base_size*sizexy,
                                      vjust = 1),
          axis.title.y = element_text(colour = "black", size = base_size*sizexy,
                                      vjust = 0))

  return(ggp)
}
```

Function `plotBlockMap`

```
#' Plot block map (workhorse plot.detectOutCell)
#'
#' @description Workhorse function for \code{plot.detectOutCell}, plotting a
#'  color coded cell map arranged by blocks
#'
#' @param x.cell a \code{matrix}
#' @param labelsx labels for the x-axis, if \code{NULL} use \code{rownames}
#' @param labelsy labels for the y-axis, if \code{NULL} use \code{colnames}
#' @param strTitle title of the blockMap
#' @param blocksize if 1 plot results for each cell of the matrix, if > 1 group
#'  neighboring cells in blocks by dimension \code{blockdim}
#' @param blockdim if \code{"row"} always group \code{blocksize} rows together,
#'  if \code{"both"} group the \code{blocksize} neighboring cells by row and
#'  column together
#' @param anglex angle of the labels on the x-axis
#' @param xtitle title for the x-axis
#' @param ytitle title for the y-axis
#' @param sizexy size of title for x-axis and y-axis
#' @param autolabel automatic labeling of combined rows and columns
#' @param base_size adjust base size of x- and y-axis labels
#'
#' @return ggp An object of class \sQuote{ggplot} containing the plotted block
#'  map
#'
#' @details Cells are visualized by color-coded squares according to the entries
#'  in the result matrix:
```

```
#'  \itemize{
#'    \item yellow (value 0) - regular cells
#'    \item red (value 1) - cellwise outliers
#'    \item blue (value 2) - bivariate cellwise outliers
#'    \item black (value 3) - rowwise outliers
#'    \item white (value 4) - missing values
#'  }
#'  Blocked cells are represented by a combined shade of neighboring colors.
#'  This method is an adapted version of the plot method proposed by Rousseeuw &
#'  Van den Bossche (2016).
#'
#' @references Rousseeuw, P.J. and Van den Bossche, W. (2016). \emph{Detecting
#'  anomalous data cells.} ArXiv e-prints
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{plot.detectOutCell}}
plotBlockMap <- function(x.cell, labelsx, labelsy, strTitle, blocksize = 1,
                         blockdim = "row", anglex = 90, xtitle = "",
                         ytitle = "", sizexy = 1.1, autolabel = TRUE,
                         base_size = 10) {

  funcSqueeze <- function(Xin, n, d, blocksize, blockdim) {
    # function for use in cellMapByBlock()
    Xblock <- matrix(0, nrow = n ,ncol = d)
    Xblockgrad <- matrix(0, nrow = n, ncol = d)

    for (i in 1:n){
      for (j in 1:d){

        if (blockdim == "both"){
        Xsel <- Xin[(1+((i-1)*blocksize)):(i*blocksize),
                    (1+((j-1)*blocksize)):(j*blocksize)]
        } else if (blockdim == "row") {
          Xsel <- Xin[(1+((i-1)*blocksize)):(i*blocksize), j]
        }

        seltable <- tabulate(unlist(Xsel), nbin = 3)
        if (blockdim == "both"){
          cnt0 <- (blocksize * blocksize) - sum(seltable)
        } else if (blockdim == "row") {
          cnt0 <- blocksize - sum(seltable)
        }

        if (sum(seltable) > 0){
          indmax <- which(seltable == max(seltable))[1]
          cntmax <- seltable[indmax]
          gradmax <- cntmax / (blocksize*blocksize)
        } else {
          indmax <- 0
          gradmax <- 1
        }

        Xblock[i,j] <- indmax
        Xblockgrad[i,j] <- gradmax
      }
    }
    return(list(X = Xblock, Xgrad = Xblockgrad))
  }

  n <- nrow(x.cell)
  d <- ncol(x.cell)

  n <- floor(n/blocksize)
  if (blockdim == "both"){
    d <- floor(d/blocksize)
  }

  # if autolabel = F, labels{x,y} will be used for the blocks.
  if (autolabel == TRUE){ # automatically combine labels for blocks
    if (blockdim == "both"){
      labx <- labelsx
      labelsx <- rep(0, d)

      for (ind in 1:d){
        labelsx[ind] <- paste(labx[(1+((ind-1)*blocksize))], "-" ,
                              labx[(ind*blocksize)], sep = "")
      }
    }

    laby <- labelsy
    labelsy <- rep(0, n)
    for (ind in 1:n){
      labelsy[ind] <- paste(laby[(1+((ind-1)*blocksize))], "-" ,
                            laby[(ind*blocksize) ])
    }
  }
  labelsy <- rev(labelsy) # will be plotted from bottom to top

  result <- funcSqueeze(x.cell, n, d, blocksize, blockdim)
  x.cell <- result$X
  Xgrad <- result$Xgrad

  # Melt data matrices for cellMap
  Xdf <- data.frame(cbind(seq(1, n, 1), x.cell))
  colnames(Xdf) <- c("row_num", seq(1, d, 1))
  rownames(Xdf) <- NULL
  Xdf$row_num <- with(Xdf, reorder(row_num, seq(n, 1, -1)))
  mX <- melt(Xdf, id.var = "row_num", value.name = "CatNr")

  Xgraddf <- data.frame(cbind(seq(1, n, 1), Xgrad))
  colnames(Xgraddf) <- c("row_num", seq(1, d, 1))
  rownames(Xgraddf) <- NULL
  Xgraddf$row_num <- with(Xgraddf, reorder(row_num, seq(n, 1, -1)))
  mXgrad <- melt(Xgraddf, id.var = "row_num", value.name = "grad")

  # Combine melted data
  mX$grad <- mXgrad$grad
```

```
  mX$rescaleoffset <- mXgrad$grad + 10*mX$CatNr
  scalerange <- c(0, 1)
  gradientends <- scalerange + rep(c(0, 10, 20, 30), each = 2)

  colorends <- c("khaki1", "khaki1", "khaki1", "red", "khaki1", "blue",
                 "khaki1", "black")

  ggp = ggplot(data = mX, aes(variable, row_num)) +
    geom_tile(aes(fill = rescale(rescaleoffset,
                                 from = range(c(0, 1) + rep(c(0, 10, 20, 30),
                                                            each = 2)))),
              colour = "white") +
    scale_fill_gradientn(colours = colorends, values = rescale(gradientends),
                         rescaler = function(x, ...) x, oob = identity) +
    ggtitle(strTitle) +
    coord_fixed() +
    theme_classic(base_size = base_size*1) +
    labs(x = xtitle, y = ytitle) +
    scale_x_discrete(expand = c(0, 0), labels = labelsx) +
    scale_y_discrete(expand = c(0, 0), labels = labelsy) +
    theme(legend.position = "none", axis.ticks = element_blank(),
          plot.title = element_text(size = base_size*2, vjust = 1,
                                    face = "bold"),
          axis.text.x = element_text(size = base_size*1.8, angle = anglex,
                                     hjust = 1, vjust = 0.5, colour = "black"),
          axis.text.y = element_text(size = base_size*1.8, angle = 0,
                                     hjust = 1, colour = "black"),
          axis.title.x = element_text(colour = "black", size = base_size*sizexy,
                                      vjust = 1),
          axis.title.y = element_text(colour = "black", size = base_size*sizexy,
                                      vjust = 0))
  return(ggp)
}
```

Function pairwLogRatios

```
#' Construct pairwise log-ratio matrix (workhorse detectDeviatingLogs and
#' detectDeviatingLogsRou)
#'
#' @description Workhorse function used in \code{detectDeviatingLogs} and
#'  \code{detectDeviatingLogsRou}, generating the pairwise log-ratio matrix
#'  between variables. Also generates the regression variables and models
#'  between two specified groups of variables.
#'
#' @param x a \code{matrix}
#' @param cols.good column indices of first group of variables
#' @param cols.bad column indices of second group of variables
#'
#' @return A list containing:
#'  \tabular{ll}{
#'    \code{cols.good} \tab column indices of first group of variables \cr
#'    \code{cols.bad} \tab column indices of second group of variables \cr
```

```r
#'   \code{y.vars} \tab names of dependent variables \cr
#'   \code{x.vars} \tab names of indipendent variables\cr
#'   \code{Y} \tab response matrix \cr
#'   \code{X} \tab regressor matrix \cr
#'   \code{Z} \tab combined matrix \code{(Y, X)}\cr
#'   \code{reg.models} \tab \code{formula}s for regression models\cr
#'   \code{rd.good} \tab names of first group of variables for distance
#'     calculation \cr
#'   \code{rd.bad} \tab names of second group of variables for distance
#'     calculation \cr
#'  }
#'
#' @details For details on the usage of the pairwise log-ratios see
#'  Beisteiner & Templ (2016).
#'
#' @references Beisteiner, L. and Templ, M. (2016). \emph{Exploratory Tools for
#'  Cellwise Outlier Detection in Compositional Data with Structural Zeros}.
#'  Unpublished master's thesis, Vienna University of Technology
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectDeviatingLogs}}
#'  \code{\link{detectDeviatingLogsRou}}
pairwLogRatios <- function(x, cols.good, cols.bad){
  l.good <- length(cols.good)
  l.bad <- length(cols.bad)

  mat.x <- data.frame(row.names = 1:nrow(x))
  mat.y <- data.frame(row.names = 1:nrow(x))
  names.mat.x <- c()
  names.mat.y <- c()
  reg.models <- NULL

  ind <- 1:(l.good-1)
  if (l.good > 1){
    for (i in ind){
      cols.new <- cols.good[-(1:i)]
      mat.x <- cbind(mat.x, log(x[, cols.good[i]]/x[, cols.new]))
      names.mat.x <- c(names.mat.x, paste0("log_x", cols.good[i], "_x",
                                           cols.new))
    }

    if (l.bad >= 1){
      ind <- 1:l.bad
      for (i in ind){
        mat.y <- cbind(mat.y, log(x[, cols.bad[i]]/x[, cols.good]))
        names.mat.y <- c(names.mat.y, paste0("log_x", cols.bad[i], "_x",
                                             cols.good))
      }
    }
  }

  if (l.bad > 1){
    ind <- 1:(l.bad-1)

    for (i in ind){
      cols.new <- cols.bad[-(1:i)]
      mat.y <- cbind(mat.y, log(x[, cols.bad[i]]/x[, cols.new]))
      names.mat.y <- c(names.mat.y, paste0("log_x", cols.bad[i], "_x",
                                            cols.new))
    }
  }

  colnames(mat.x) <- names.mat.x
  colnames(mat.y) <- names.mat.y

  # 1) bad/good vs. good/good ratios
  # 2) bad/bad vs. good/good ratios
  if (l.good > 1){
    reg.x <- paste0(" ~ ", paste0(names.mat.x, collapse = " + "))
    reg.models <- lapply(names.mat.y, function(x){
      paste0(x, reg.x, sep = "")
    })
  }

  # 3) robust distances between all good ratios
  rd.good <- names.mat.x
  rd.bad <- NULL

  # 4) robust distances between all bad ratios
  if (l.bad > 1){
    ind <- 1:(l.bad-1)

    for (i in ind){
      cols.new <- cols.bad[-(1:i)]
      rd.bad<- c(rd.bad, paste0("log_x", cols.bad[i], "_x", cols.new))
    }
  }

  return(list(cols.good = cols.good, cols.bad = cols.bad,
              y.vars = names.mat.y, x.vars = names.mat.x, Y = mat.y,
              X = mat.x, Z = cbind(mat.y, mat.x), reg.models = reg.models,
              rd.good = rd.good, rd.bad = rd.bad))
}
```

Function `detectUnivOut`

```r
#' Detect univariate outliers (workhorse detectDeviatingLogs)
#'
#' @description Workhorse function used in \code{detectDeviatingLogs},
#'  performing the initial univariate outlier detection step.
#'
#' @param x a \code{matrix}
#'
#' @return u a 0/1-matrix, indicating 1 = univariate outliers, 0 = regular cells
#'
```

```
#' @details This method is an adapted version of the univarate outlier detection
#'  method proposed by Rousseeuw & Van den Bossche (2016).
#'
#' @references Beisteiner, L. and Templ, M. (2016). \emph{Exploratory Tools for
#'  Cellwise Outlier Detection in Compositional Data with Structural Zeros}.
#'  Unpublished master's thesis, Vienna University of Technology
#' @references Rousseeuw, P.J. and Van den Bossche, W. (2016). \emph{Detecting
#'  anomalous data cells.} ArXiv e-prints
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectDeviatingLogs}}
detectUnivOut <- function(x){

  # Univariate estimators of location and scale
  loc1StepM <- function(x, c1 = 3, precScale = 1e-12){
    # Computes the first step of an algorithm for
    # a location M-estimator using the biweight.
    # Note that c1 is expressed in unnormalized MAD units.
    # In the usual units it is thus c1/qnorm(3/4).
    medx <- median(x)
    ax <- abs(x - medx)
    denom <- c1 * median(ax)
    mu <- if(denom > precScale) {
      ax <- ax/denom
      w <- 1 - ax * ax
      w <- ((abs(w) + w)/2)^2
      sum(x * w)/sum(w)
    }
    else{
      medx
    }

    return(mu)
  }

  scale1StepM <- function(x, c2 = 2.5, delta = 0.844472, precScale = 1e-12){
    # Computes the first step of an algorithm for
    # a scale M-estimator using the Huber rho.
    # Assumes that the univariate data in x has already
    # been centered, e.g. by subtracting loc1StepM.
    # Note that c2 is expressed in unnormalized MAD units.
    # In the usual units it is thus c2/qnorm(3/4).
    # If you change c2 you must also change delta.
    n <- length(x)
    sigma0 <- median(abs(x))
    if(c2*sigma0 < precScale){
      return(sigma0)
    } else {
      x <- x/sigma0
      rho <- x^2
      rho[rho > c2^2] <- c2^2
      return(sigma0 * sqrt(sum(rho)/(n*delta)))
    }
```

```r
  }

  detectUniv <- function(x, cut.val){
    # Detects outliers and sets them to NA.
    # Assumes that the data have already been standardized.
    x.out <- rep(0, length(x))
    x.out[(abs(x) > cut.val)] <- 1
    return(x.out)
  }

  # compute ratios
  x <- sapply(1:ncol(x), function(i){
    x[, i] <- x[, i]/rowSums(x[,-i])
  })

  # Robust standardization
  locX <- apply(x, 2, loc1StepM)
  z <- sweep(x, 2, locX)
  scaleX <- apply(z, 2, scale1StepM)
  z <- sweep(z, 2, scaleX, "/")

  cut.val <- sqrt(qchisq(0.99,1))

  u <- apply(z, 2, detectUniv, cut.val = cut.val)
  return(u)
}
```

Function checkDataFrame

```r
#' Check data frame before MD calculation (workhorse detectOutCell)
#'
#' @description Workhorse function used in \code{detectOutCell}, checking
#'  whether variables are suitable for MD calculation (\code{factor} variables
#'  or variables that are too discrete are identified and ruled out)
#'
#' @param x a \code{matrix}
#' @param reg.vars variables containing the compositional data
#' @param numDiscrete a column that takes on \code{numDiscrete} or fewer values
#'  will be considered discrete and not used in the analysis.
#' @param precScale only consider columns whose scale is > \code{precScale}.
#'  Here scale is measured by the median absolute deviation.
#' @param suppressOutput suppress additional output provided by the algorithms
#'
#' @return colMHD \code{TRUE}/\code{FALSE} vector, indicating which variables
#'  are suitable for MD calculation
#'
#' @details This method is an adapted version of the method proposed by
#'  Rousseeuw & Van den Bossche (2016).
#'
#' @references Beisteiner, L. and Templ, M. (2016). \emph{Exploratory Tools for
#'  Cellwise Outlier Detection in Compositional Data with Structural Zeros}.
#'  Unpublished master's thesis, Vienna University of Technology
#' @references Rousseeuw, P.J. and Van den Bossche, W. (2016). \emph{Detecting
#'  anomalous data cells.} ArXiv e-prints
#'
#' @author Lukas Beisteiner, Matthias Templ
#' @note License: GPL-2
#' @seealso \code{\link{detectOutCell}}
checkDataFrame <- function(x, reg.vars, numDiscrete = 3, precScale = 1e-12,
                           suppressOutput = FALSE){

  wnq <- function(string, qwrite = 1){
    # auxiliary function
    # writes a line without quotes
    if(qwrite == 1) write(noquote(string), file = "", ncolumns = 100)
  }

  pnq <- function(string, qwrite = 1){
    # auxiliary function
    # prints a line without quotes
    if(qwrite == 1) print(noquote(string))
  }

  # +-------------------------------------------------------------------+
  # | 0) Initial checks                                                 |
  # +-------------------------------------------------------------------+
  if(!all(x[, reg.vars] >= 0, na.rm = TRUE)){
    stop("Compositional variables of x must be greater or equal to zero!")
  }

  if (is.null(dim(x))){
    stop("x must be a data.frame, data.table or matrix")
  }

  # +-------------------------------------------------------------------+
  # | 1) Deselect non-numeric variables                                 |
  # +-------------------------------------------------------------------+
  colNumeric <- sapply(x, is.numeric)
  sumNumeric <- sum(colNumeric)

  colNotNumeric <- colNumeric == FALSE
  sumNotNumeric <- sum(colNotNumeric)

  if(sumNotNumeric > 0) {
    if (suppressOutput == FALSE){
      wnq(" ")
      wnq(paste("The input data contained ", sumNotNumeric,
                " non-numeric columns (variables).", sep=""))
      wnq("Their column names are:")
      wnq("")
      pnq(colnames(x)[colNotNumeric])
      wnq(" ")
    }
    if(sumNumeric > 1) {
      if (suppressOutput == FALSE){
        wnq(paste("These columns will be ignored for calculating the robust ",
                   "Mahalanobis distances.", sep = ""))
        wnq(paste("We continue the calculation with the remaining ",
                   sumNumeric, " numeric columns:", sep=""))
      }
      x <- x[, colNumeric, drop = FALSE]
    } else {
      if(sumNumeric == 0) stop(" No columns remain, stopping procedure.")
      if(sumNumeric == 1) stop("Only 1 column remains, stopping procedure.")
    }
    wnq(" ")
    pnq(names(which(colNumeric)))
  }

  colMHD <- colNumeric

  # +-----------------------------------------------------------------------+
  # | 2) Deselect discrete variables, loosely defined as variables that take |
  # |    on numDiscrete or fewer values, such as binary variables        |
  # +-----------------------------------------------------------------------+
  valueCount <- apply(x, 2, function(xj){
    sum(!is.na(unique(xj)))
  })

  colNotDiscrete <- (valueCount > numDiscrete)
  sumNotDiscrete <- sum(colNotDiscrete)
  colDiscrete <- colNotDiscrete == F
  sumDiscrete <- sum(colDiscrete)

  if(sumDiscrete > 0) {
    if (suppressOutput == FALSE){
      wnq(" ")
      wnq(paste("The data contained ", sumDiscrete," discrete columns with ",
                sumDiscrete," or fewer values.",sep=""))
      wnq("Their column names are:")
      wnq(" ")
      pnq(colnames(x)[colDiscrete])
      wnq(" ")
    }
    if(sumNotDiscrete > 1) {
      if (suppressOutput == FALSE){
        wnq(paste("These columns will be ignored for calculating the robust ",
                   "Mahalanobis distances.", sep = ""))
        wnq(paste("We continue with the remaining ", sumNotDiscrete, " columns:"
                   , sep = ""))
      }
      x <- x[, colNotDiscrete, drop = FALSE]
    } else {
      if(sumNotDiscrete == 0) stop("No columns remain, stopping procedure.")
      if(sumNotDiscrete == 1) stop("Only 1 column remains, stopping procedure.")
    }

    colMHD[colMHD == TRUE] <- colNotDiscrete
    wnq(" ")
    pnq(names(which(colMHD)))
  }

  # +----------------------------------------------------------------------------+
  # | 3) Deselect columns for which the median absolute deviation is zero. |
  # |    This is equivalent to saying that 50% or more of its values are |
  # |    equal.                                                             |
  # +----------------------------------------------------------------------------+
  colScale <- apply(x, 2, mad, na.rm = TRUE)
  colGood <- colScale > precScale
  sumGood <- sum(colGood)
  colBad <- colGood == FALSE
  sumBad <- sum(colBad)

  if(sumBad > 0) {
    if (suppressOutput == FALSE){
      wnq(" ")
      wnq(paste("The data contained ", sumBad," columns with zero or tiny median",
                " absolute deviation.",sep=""))
      wnq("Their column names are:")
      wnq(" ")
      pnq(colnames(x)[colBad])
      wnq(" ")
    }
    if(sumGood > 1) {
      if (suppressOutput == FALSE){
        wnq(paste("These columns will be ignored for calculating the robust ",
                  "Mahalanobis distances.", sep = ""))
        wnq(paste("We continue with the remaining ", sumGood, " columns:",
                  sep = ""))
      }
      x <- x[, colGood, drop = FALSE]
    } else {
      if(sumGood == 0) stop("No columns remain, stopping procedure.")
      if(sumGood == 1) stop("Only 1 column remains, stopping procedure.")
    }

    colMHD[colMHD == TRUE] <- colGood
    if (suppressOutput == FALSE){
      wnq(" ")
      pnq(names(which(colMHD)))
    }
  }

  return(colMHD = colMHD)
}
```

Bibliography

J. Aitchison. *The Statistical Analysis of Compositional Data Monographs on Statistics and Applied Probability.* Chapman & Hall Ltd., London (UK), 1986.

J. Bacon-Shone. *A. Buccianti, G. Mateu-Figueras and V. Pawlowsky-Glahn (eds): Compositional Data Analysis in the Geosciences: From Theory to Practice*, volume 22. Springer-Verlag, 2008.

J.A. Cortes. On the Harker Variation Diagrams; A Comment on "The Statistical Analysis of Compositional Data. Where Are We and Where Should We Be Heading?" by Aitchison and Egozcue (2005). *Mathematical Geosciences*, 41(7):817–828, 2009.

C. Croux and G. Haesbroeck. Influence function and efficiency of the minimum covariance determinant scatter matrix estimator. *Journal of Multivariate Analysis*, 71(2):161 – 190, 1999.

U.S. State Department. *Albania 2014 International Religious Freedom Report.* Bureau of Democracy Human Rights and Labor, United States Department of State., 2014.

D.L. Donoho and P.J. Huber. The Notion of Breakdown Point. *A Festschrift for Erich Lehmann, edited by P. Bickel, K. Doksum, and J.L. Hodges Jr. Wadsworth*, 1983.

J.J. Egozcue and V. Pawlowsky-Glahn. Groups of Parts and Their Balances in Compositional Data Analysis. *Mathematical Geology*, 37(7):795–828, 2005.

J.J. Egozcue, V. Pawlowsky-Glahn, G. Mateu-Figueras, and C. Barcelo-Vidal. Isometric Logratio Transformations for Compositional Data Analysis. *Mathematical Geology*, 35(3):279–300, 2003.

P. Filzmoser, R. Maronna, and M. Werner. Outlier identification in high dimensions. *Comput. Stat. Data Anal.*, 52(3):1694–1711, January 2008. ISSN 0167-9473.

R. Gnanadesikan and J.R. Kettenring. Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, 28(1):81–124, 1972.

F.R. Hampel. A General Qualitative Definition of Robustness. *Ann. Math. Statist.*, 42(6):1887–1896, 12 1971.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

D.M. Hawkins, Dan Bradu, and G.V. Kass. Location of several outliers in multiple-regression data using elemental sets. *Technometrics*, 26(3):197–208, 8 1984.

K. Hron, M. Templ, and P. Filzmoser. Imputation of Compositional Data Using Robust Methods. Research report sm-2008-4, Department of Statistics and Probability Theory, Vienna University of Technology, 2008.

K. Hron, M. Templ, and P. Filzmoser. Exploratory Tools for Outlier Detection in Compositional Data with Structural Zeros. 2015. Submitted for publication.

R.A. Johnson and D.W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall series in statistics. Prentice Hall, 1998.

M. Koller and W.A. Stahel. Sharpening Wald-Type Inference in Robust Regression for Small Samples. *Computational Statistics and Data Analysis*, 55(8):2504 – 2515, 2011.

N. Locantore, J.S. Marron, D.G. Simpson, N. Tripoli, J.T. Zhang, K.L. Cohen, G. Boente, R. Fraiman, B. Brumback, C. Croux, J. Fan, A. Kneip, J.I. Marden, D. Peña, J. Prieto, J.O. Ramsay, M.J. Valderrama, and A.M. Aguilera. Robust principal component analysis for functional data. *Test*, 8(1):1–73, 1999.

R.A. Maronna, D.R. Martin, and V.J. Yohai. *Robust Statistics: Theory and Methods*. Wiley Series in Probability and Statistics. Wiley, 2006.

G. Pison, S. Van Aelst, and G. Willems. Small sample corrections for lts and mcd. *Metrika*, 55(1):111–123, 2002.

D.M. Rocke. Robustness properties of s-estimators of multivariate location and shape in high dimension. *Ann. Statist.*, 24(3):1327–1345, 06 1996.

P.J. Rousseeuw. Least Median of Squares Regression. *Journal of The American Statistical Association*, 79:871–880, 1984.

P.J. Rousseeuw. Multivariate estimation with high breakdown point. *Mathematical statistics and applications*, 8:283–297, 1985.

P.J. Rousseeuw and W. Van den Bossche. Detecting anomalous data cells. *ArXiv e-prints*, January 2016.

C. Stewart and C. Field. Managing the essential zeros in quantitative fatty acid signature analysis. *Journal of Agricultural, Biological, and Environmental Statistics*, 16(1):45–69, 2010.

M. Templ, P. Filzmoser, and K. Hron. *Imputation Methods in robCompositions*, 2015. R package version 1.9.1.

V.J. Yohai. High Breakdown-Point and High Efficiency Robust Estimates for Regression. *Ann. Statist.*, 15(2):642–656, 06 1987.