# SGB multivariate regression

Monique Graf
Institut de statistique
Université de Neuchâtel, Switzerland

May 9, 2019

**Abstract**

The main features of the R-package **SGB** are described. A new distribution on the simplex is defined, the Simplicial Generalized Beta distribution (SGB). A regression procedure based on the SGB for compositions as dependent vectors is set up. Package **SGB** offers an imputation method for missing sub-compositions, graphical diagnostic checks and goodness of fit tests. A stepwise regression for backward elimination permits to simplify the model.

**Key words:** Probability Distributions, Multivariate Techniques, Regression, Statistical Inference.

## 1 Introduction

Compositions are multivariate constrained data. Each observation is a positive vector with constant sum (it is said to be *closed*) and this peculiarity requires special analytical methods. A seminal reference is Aitchison (1986) who developed tools based on log-ratios of parts and the logistic normal distribution, i.e. ratios of parts are log-normally distributed. A quick introduction to the geometry in the simplex is given e.g. by Egozcue and Pawlovsky-Glahn (2011). An alternative to the logistic normal is the Dirichlet distribution, see a modern account in Ng et al. (2011). Hijazi and Jernigan (2009) allow the Dirichlet parameters to depend on auxiliary variables and Gueorguieva et al. (2008) propose several diagnostic procedures. Monti et al. (2011) defined the scaled Dirichlet by introducing a constant scale composition. The Beta distribution is a special case of Dirichlet for two-parts compositions.

Several R-packages exist in the topic of compositions: the general purpose R-package **compositions** (van den Boogaart et al., 2014) gives the possibility to fit a Dirichlet distribution without covariates. The R-package **robCompositions** (Templ et al., 2011) specializes on robust methods and offers only the logistic normal when a distribution needs to be specified. The package **betareg** (Grün et al., 2012) proposes regression models based on the Beta distribution, where both the scale composition and the shape parameters can be modeled with explanatory variables. Finally, Dirichlet regressions for the shape parameters are proposed in **DirichletReg** (Maier, 2015).

In the present R-package **SGB**, a more flexible generalization of the Dirichlet distribution is developed. It is called the *simplicial generalized Beta* (SGB). Craiu and Craiu (1969), see (Kotz et al., 2000, p. 490), obtained the density and called it "generalized Dirichlet". The SGB parameters encompass an overall shape parameter $a$, a scale composition $\mathbf{b}$, and a vector of Dirichlet shape parameters $\mathbf{p}$.

In the Dirichlet and the scaled Dirichlet regressions, the shape parameters are modeled with auxiliary variables. On the other hand, in the ordinary logistic normal regression, it is the scale composition that is made dependent on auxiliary variables. The modeling of scales seems easier to interpret than the modeling of shapes. Thus in the SGB regression:

- The scale composition are modeled in the same way as for the logistic normal regression, i.e. each auxiliary variable generates $D-1$ parameters, where $D$ is the number of parts.
- The $D$ Dirichlet shape parameters, one for each part in the compositions, are estimated as well.
- An additional overall shape parameter is introduced in the SGB.
- Use of survey weights is an option.
- A backward stepwise regression procedure permits to eliminate non significant regression parameters.
- Imputation of missing parts is possible.

The SGB distribution is defined in Section 2. In Section 3, the SGB regression procedure is briefly described. In Section 4, examples of the main features of the package are given.

# 2 SGB distribution

The Dirichlet distribution can be viewed as the distribution of $\mathbf{U} = \mathcal{C}(\mathbf{Y})$, where $\mathbf{Y} = (Y_j)_{j=1,...,D}$ is a vector of independent $Gamma(p_j)$ components and $\mathcal{C}(.)$ is the closure operation (i.e. $U_j = Y_j / \sum_{i=1}^{D} Y_i$). The SGB distribution follows the same construction, with the Gamma distribution replaced by the generalized Gamma, that is the underlying $Y_i$ are independent $GG(a, c\, b_j, p_j)$, $c$ being an arbitrary positive constant. The parameters are all positive and $\mathbf{b} = (b_1, ..., b_D)$ is itself a composition, the *scale composition*. The SGB can also be generated from the Dirichlet

**Definition** *Suppose that* $\mathbf{Z} = (Z_1, ..., Z_D)$ *follows a Dirichlet*$(p_1, ..., p_D)$ *distribution. Then the random composition* $\mathbf{U} = (U_1, ..., U_D)$, $(D \geq 2)$, *given by*

$$U_j = \frac{b_j\, Z_j^{1/a}}{\sum_{i=1}^{D} b_i\, Z_i^{1/a}}, j = 1, ..., D \quad or \quad \mathbf{U} = \mathcal{C}[\mathbf{b}\, \mathbf{Z}^{1/a}]$$

*follows a* $SGB(a, \{b_j, p_j, j = 1, ..., D\})$ *distribution.*

*All parameters are positive; a is an overall shape parameter,* $\mathbf{b} = (b_1, ..., b_D)$ *a scale composition and* $\mathbf{p} = (p_1, ..., p_D)$ *the vector of Dirichlet shape parameters.*

Conversely, the random composition $\mathbf{Z}$ can be written in function of $\mathbf{U}$,

$$Z_j = \frac{(U_j/b_j)^a}{\sum_{i=1}^{D} (U_i/b_i)^a}, j = 1, ..., D \quad \text{or} \quad \mathbf{Z} = \mathcal{C}[(\mathbf{U}/\mathbf{b})^a]. \tag{1}$$

Because $U_D = 1 - \sum_{j=1}^{D-1} U_j$, there are only $D-1$ variables in the composition $\mathbf{U}$. The $L_a$-norm of the vector $(\mathbf{u}/\mathbf{b})$ is

$$\|\mathbf{u}/\mathbf{b}\|_a \quad = \quad \left[ \sum_{k=1}^{D-1} (u_k/b_k)^a + \left( (1 - \sum_{j=1}^{D-1} u_j)/b_D \right)^a \right]^{1/a}.$$

The probability density of the $SGB(a, \{b_j, p_j, j = 1, ..., D\})$ distribution is obtained as

$$f_{\mathbf{U}}(\mathbf{u}_{-D}) = \frac{\Gamma(P)a^{D-1}}{\prod_{j=1}^{D}\Gamma(p_j)} \prod_{k=1}^{D-1} \left\{ \frac{u_k/b_k}{\|\mathbf{u}/\mathbf{b}\|_a} \right\}^{ap_k} \left\{ \frac{\left(1 - \sum_{j=1}^{D-1} u_j\right)/b_D}{\|\mathbf{u}/\mathbf{b}\|_a} \right\}^{ap_D} \frac{1}{\prod_{k=1}^{D-1} u_k \left(1 - \sum_{j=1}^{D-1} u_j\right)}, \quad (2)$$

$$u_k > 0,\ k = 1, ..., D-1, \qquad 1 - \sum_{j=1}^{D-1} u_j > 0.$$

Consider the so called Aitchison's expectation, i.e. the image in the simplex of the expectation of centered log-composition,

$$\hat{\mathbf{U}} = \mathrm{E}_A(\mathbf{U}) = \mathcal{C}\{\exp\{\mathrm{E}\log[\mathbf{U}/g(\mathbf{U})]\}\},$$

In the SGB context, with $\psi(.)$ the digamma function, one obtains

$$\mathrm{E}_A(U_k) = \frac{b_k \exp\{\psi(p_k)/a\}}{\sum_{j=1}^{D} b_j \exp\{\psi(p_j)/a\}} \quad k = 1, ..., D. \quad (3)$$

The fitted composition $\hat{\mathbf{U}}$ is defined as the estimated value of $\mathrm{E}_A(\mathbf{U})$ obtained by plugging estimated parameters into the formula.

Consider a random composition $\mathbf{U}$ following a SGB distribution with known parameters. Suppose that only a sub-composition $\mathbf{v}$ of a realization $\mathbf{u}$ is observed, and let $\mathbf{w}$ be the sub-composition with the missing parts. We have $\mathbf{u} = (x\mathbf{v}, (1-x)\mathbf{w})$, $0 < x < 1$, realization of $\mathbf{U} = (X\mathbf{V}, (1-X)\mathbf{W})$, $0 < X < 1$. One can show that $X$ follows a SGB distribution as well, and that the random sub-compositions $\mathbf{V}$ and $\mathbf{W}$ are independent (Graf, 2017, Theorem 4). The best guess for $\mathbf{w}$ is thus given by $\widehat{\mathbf{w}} = \mathrm{E}_A(\mathbf{W})$ (see Equation 3). Then the imputed composition is

$$\mathrm{E}_A(\mathbf{U}|\mathbf{v}) \doteq \widehat{\mathbf{u}} = (\widehat{x}_{\mathbf{v}}\mathbf{v}\,,\, (1 - \widehat{x}_{\mathbf{v}})\widehat{\mathbf{w}})\,, \quad (4)$$

where $\widehat{x}_{\mathbf{v}} = \mathrm{E}_A\,(\,X|\,\mathbf{V} = \mathbf{v}, \mathbf{W} = \widehat{\mathbf{w}})$, see (Graf, 2017, Theorem 4 (5)). The imputed composition can be viewed as the *conditional Aitchison's expectation given the sub-composition* $\mathbf{v}$. Equation (4) extends to the case of a totally missing composition and is then identical to Equation (3).

# 3 SGB regression model

In the R-package **SGB** regression models can be set up for the scale composition $\mathbf{b}$. The shape parameters $a$ and $\mathbf{p}$ are estimated as well, but are supposed constant across compositions.

## 3.1 Model

The SGB regression models follow the principles of log-ratio analysis advocated by Aitchison (1986). We define a general $D \times (D-1)$ contrast matrix $\mathbf{V}$, such that

$$\mathbf{1}_D^t \mathbf{V} = \mathbf{0}_{D-1}^t,$$

where $\mathbf{1}_D$ is a $D$-vector of ones and $\mathbf{0}_{D-1}$ is a $(D-1)$-vector of zeros. The model for scales is the general linear model. Let $\mathbf{X}$ be a $n \times p$ matrix of explanatory variables, where $n$ is the sample size. Let $\mathbf{u}_i, i = 1, ..., n$ be the composition associated to $\mathbf{x}_i^t$, the i-th row of $\mathbf{X}$. Then the scales are modeled by

$$\log(\mathbf{b}_i^t)\mathbf{V} = \mathbf{x}_i^t \mathbf{B}, \quad (5)$$

where

$$\mathbf{B} = \left(\boldsymbol{\beta}_1 ... \boldsymbol{\beta}_{D-1}\right)$$

is the $p \times (D-1)$ - matrix of regression parameters for the $(D-1)$ contrasts, columns of $\log(\mathbf{u}_i^t)\mathbf{V}, i = 1, ..., n$.

### 3.2 Fitting procedure

There is the possibility to introduce sampling weights into the procedure. These weights $w_i, i = 1, ..., n$ are scaled to sum to $n$.

The pseudo-log-likelihood is the weighted version of the log-likelihood and is given by

$$\begin{aligned} & \ell\left(a, (b_1, p_1), ..., (b_D, p_D) | \mathbf{u}_{i,-D}, i = 1, ..., n\right) \\ = \; & n\left[(D-1)\log(a) + \log\Gamma(P) - \sum_{k=1}^{D}\log\Gamma(p_k)\right] + \sum_{i=1}^{n} w_i \sum_{k=1}^{D} p_k \log z_k(\mathbf{u}_i) \\ & - \text{terms not depending on parameters.} \end{aligned}$$

with $z(\mathbf{u}_i) = (z_1(\mathbf{u}_i), ..., z_D(\mathbf{u}_i))$ given at Equation (1) and $P = \sum_{j=1}^{D} p_j$.

The model is estimated by maximizing the pseudo-likelihood using a constrained optimization method, the augmented Lagrangian, see e.g. Madsen et al. (2004), and implemented in the R-package **alabama** as function `auglag` (Varadhan, 2015). The gradient is computed analytically and the Hessian numerically. The default constraints are

$$\begin{aligned} a &> 0.1 \quad \text{(to avoid numerical problems)} \\ p_j &> 0, \quad j = 1, ..., D \\ a\, p_j &> \texttt{bound}, \quad \text{by default, } \texttt{bound} = 2.1. \end{aligned}$$

Moments of ratios of parts following the SGB distribution only exist up to $a\, p_j$. Thus `bound` $= 2.1$ guarantees the existence of variances of all ratios of parts. Notice that the most important variables, the log-ratios of parts, possess moment of all orders.

### 3.3 Initial values

The initial values of the regression parameters $\mathbf{B}$ are estimated by multiple linear regression using the weights if specified. Several proposals are found for the choice of initial values when fitting a Dirichlet model. The method by Wicker et al. (2008) has been adapted to the SGB for the initial values of the Dirichlet shape parameters $p_1, ..., p_D$. The overall shape parameter $a$ must be chosen by the user.

A very handy feature of `alabama::auglag` is that the initial values need not satisfy the constraints, and that general (twice derivable) constraints on parameters can be introduced. The price to pay is the speed.

## 4 Examples

### 4.1 Regression with arbitrary log-ratio transform

In the data set `carseg` (Morais and Thomas-Agnan, 2019) segment shares of car sales in five categories (SA, SB, ... ,SE) according to the size of the car chassis during 152 months, as well as seven economic variables are given. The file with compositions is `uc`. Multiplying the log-compositions by the matrix `Vc`, we obtain the log-ratios between two adjacent categories. It is important to name the columns of `Vc`; they will be used to document the parameter estimates and to specify the model.

```
library(SGB)
data(carseg)
## Extract the compositions
uc <- as.matrix(carseg[,(1:5)])
## Define the log-ratio transformation matrix
```

```
Vc <- matrix(c( 1,0,0,0,
               -1,1,0,0,
               0,-1,1,0,
               0,0,-1,1,
               0,0,0,-1),ncol=4,byrow=TRUE)
colnames(Vc) <- c("AB","BC","CD","DE")
rownames(Vc) <- colnames(uc)
print(Vc)
```

```
##    AB BC CD DE
## SA  1  0  0  0
## SB -1  1  0  0
## SC  0 -1  1  0
## SD  0  0 -1  1
## SE  0  0  0 -1
```

All seven explanatory variables (see the help on `carseg`) are introduced in a meaningful way into the `Formula`: positive variables are changed to log; `PAC` is the indicator of an incentive period during which people were encouraged to change their vehicle. The syntax for `Formula` stems from the `R`-package **Formula** (Varadhan, 2015) and extends the ordinary `formula` to a multivariate dependent vector. In the `R`-package **SGB** the left hand side of the formula defines the same model for all log-ratio transforms (left hand side of Equation 5). There is the possibility to define different models for each lrt by fixing some coefficients to zero (see Section 4.3).

```
## Fit a model
Form1 <- Formula(AB | BC | CD | DE ~  log(expend) + I(PAC*log(expend)) +
         log(sent) + log(FBCF) + log(price) + rates)
obj1 <- regSGB(Form1, data=list(carseg, uc, Vc), shape10 = 4.4,
               control.outer = list(trace=FALSE))
```

By default the algorithm prints all iterations of the outer loop until convergence (here `control.outer = list(trace=FALSE)` suppresses the output). The `print` method for `regSGB` objects gives the call and the estimated parameters.

```
obj1
```

```
## Call:
## regSGB.formula(Formula = Form1, data = list(carseg, uc, Vc),
##     shape10 = 4.4, control.outer = list(trace = FALSE))
##
## Parameters:
##  [1]  2.358762e+00 -2.349740e+00 -1.018023e+01 -2.450899e+01 -2.181868e+01
##  [6]  8.263274e-01  6.486686e-01  2.835086e+00  3.336927e+00  4.724404e-02
## [11]  1.290497e-02  2.869386e-05  1.362196e-02 -5.979646e-01  2.894633e-01
## [16]  1.220414e-01 -6.866762e-01 -6.511490e-01  1.787577e-01 -8.168048e-01
## [21] -1.362363e+00  5.286386e-01 -3.740183e-01  2.255387e-01  1.264031e-01
## [26]  1.042212e-01  1.816377e-02 -3.182095e-02  4.741377e-02  1.603896e+01
## [31]  2.892991e+01  3.591596e+01  1.556963e+01  7.754796e+00
```

`summary` is a list with 3 components, the call, the formula and the estimated parameters with two types of standard deviation. Notice that the shape parameters are defined in the compositional space (the simplex), whereas the regression parameters act on log-ratio transforms. The standard deviations `StdErr1` are based on the inverse of the Hessian obtained numerically with `R`-package **numDeriv** (Gilbert

and Varadhan, 2016). `StdErr2` is the sandwich estimate (Huber, 1967) which is robust to some departures of the model. In the example, we see that the robust estimate is generally larger than the classical one. The p-values are based on the asymptotic normal distribution for the parameter estimates.

```
## Results
summary(obj1)


## $call
## regSGB.formula(Formula = Form1, data = list(carseg, uc, Vc),
##     shape10 = 4.4, control.outer = list(trace = FALSE))
##
## $Formula
## AB | BC | CD | DE ~ log(expend) + I(PAC * log(expend)) + log(sent) +
##     log(FBCF) + log(price) + rates
##
## $parameters
##                  Parameters Initial Estimate StdErr1 StdErr p.value signif
## 1                     shape1   4.400    2.359   0.075  0.082   0.000    ***
## 2           (Intercept).AB  -1.464   -2.350   0.710  0.932   0.012      *
## 3           (Intercept).BC -10.512  -10.180   1.519  1.997   0.000    ***
## 4           (Intercept).CD -23.778  -24.509   1.082  1.431   0.000    ***
## 5           (Intercept).DE -20.484  -21.819   0.429  0.565   0.000    ***
## 6           log(expend).AB   0.706    0.826   0.312  0.294   0.005     **
## 7           log(expend).BC   0.638    0.649   0.276  0.299   0.030      *
## 8           log(expend).CD   2.780    2.835   0.310  0.277   0.000    ***
## 9           log(expend).DE   3.187    3.337   0.417  0.380   0.000    ***
## 10 I(PAC * log(expend)).AB   0.047    0.047   0.003  0.003   0.000    ***
## 11 I(PAC * log(expend)).BC   0.013    0.013   0.003  0.002   0.000    ***
## 12 I(PAC * log(expend)).CD   0.000    0.000   0.003  0.003   0.993
## 13 I(PAC * log(expend)).DE   0.012    0.014   0.005  0.004   0.000    ***
## 14            log(sent).AB  -0.641   -0.598   0.139  0.132   0.000    ***
## 15            log(sent).BC   0.296    0.289   0.114  0.110   0.009     **
## 16            log(sent).CD   0.124    0.122   0.139  0.114   0.283
## 17            log(sent).DE  -0.679   -0.687   0.199  0.164   0.000    ***
## 18            log(FBCF).AB  -0.603   -0.651   0.408  0.367   0.076      .
## 19            log(FBCF).BC   0.216    0.179   0.327  0.311   0.566
## 20            log(FBCF).CD  -0.789   -0.817   0.395  0.350   0.020      *
## 21            log(FBCF).DE  -1.290   -1.362   0.556  0.500   0.006     **
## 22          log(price).AB   0.531    0.529   0.125  0.123   0.000    ***
## 23          log(price).BC  -0.382   -0.374   0.112  0.113   0.001    ***
## 24          log(price).CD   0.229    0.226   0.125  0.117   0.053      .
## 25          log(price).DE   0.139    0.126   0.171  0.148   0.393
## 26               rates.AB   0.101    0.104   0.019  0.018   0.000    ***
## 27               rates.BC   0.017    0.018   0.015  0.015   0.211
## 28               rates.CD  -0.034   -0.032   0.018  0.015   0.040      *
## 29               rates.DE   0.043    0.047   0.026  0.021   0.021      *
## 30              shape2.SA   5.200   16.039   1.501  1.521   0.000    ***
## 31              shape2.SB   5.200   28.930   0.096  0.098   0.000    ***
## 32              shape2.SC   5.200   35.916   0.106  0.108   0.000    ***
## 33              shape2.SD   5.200   15.570   1.587  1.649   0.000    ***
## 34              shape2.SE   5.200    7.755   1.024  1.050   0.000    ***
##
## $signif.codes
## [1] " 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1"
##
```

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

`table.regSGB` is a data-frame with the overall results for the fitted object.

```
## Overall results
t1 <- table.regSGB(obj1)
print(t1)


##                   statistics
## value            401.4176523
## n.par             34.0000000
## n.par.fixed        0.0000000
## AIC             -734.8353045
## Rsquare            0.8649558
## convergence        0.0000000
## kkt1               1.0000000
## kkt2               1.0000000
## counts.function  376.0000000
## counts.gradient  102.0000000
```

`value` is the value of the log-likelihood at the last iteration, `n.par` is the number of parameters and `n.par.fixed` the number of fixed parameters (see Example 4.4); `AIC` is Akaike's criterion, `Rsquare` is the ratio of the total variance of the fitted compositions to the observed compositions (see below). The next parameters describe the run: `convergence` is 0 if the algorithm converged, `kkt1` is 1, if the first Karush-Kuhn-Tucker condition is met (i.e. the gradient of the log-likelihood is close to zero), `kkt2` is 1, if the second Karush-Kuhn-Tucker condition is met (i.e. the Hessian is negative definite), `counts.function` (`counts.gradient`) is the number of times the log-likelihood (the gradient) was evaluated.

`Rsquare` (Hijazi and Jernigan, 2009) is a statistic similar to the ordinary $R^2$ adapted to variables in the simplex. It is based on Aitchison's total variation (Aitchison, 1986), see also Egozcue and Pawlovsky-Glahn (2011). More precisely, let $\mathbf{U}$ be a random composition. Consider the following covariance matrix

$$\mathbf{\Gamma} = \mathbf{\Gamma}(\mathbf{U}) = (\gamma_{ij})_{i,j=1,...,D} = \text{Cov}\left[\log\left(\frac{\mathbf{U}}{g(\mathbf{U})}\right)\right],$$

where $g(\mathbf{U})$ is the geometric mean of the parts of $\mathbf{U}$. The total variation of a composition $\mathbf{U}$ is given by

$$\text{totvar}(\mathbf{U}) = \text{tr}(\mathbf{\Gamma}) = \sum_{i=1}^{D} \gamma_{ii}.$$

The fitted composition $\hat{\mathbf{U}}$ is defined as the estimated value of the Aitchison's expectation (see Equation 3). Replacing the parameters by their estimated values, the fitted Aitchison's expectations under the SGB model are obtained with function `meanAobj.SGB`. Then `Rsquare` is given by

$$\frac{\text{totvar}(\hat{\mathbf{U}})}{\text{totvar}(\mathbf{U})}.$$

Some words of caution on the interpretation of `Rsquare`: there is no anova table in the SGB (and Dirichlet) context. It can happen that adding parameters decreases `Rsquare`, which is impossible in ordinary multiple linear regression. Another point is that `Rsquare` does not depend on a concept like the degrees of freedom.

The z-transforms of parts $\mathbf{z} = (z_1, ..., z_D)$ were defined in Equation (1). Under the $SGB(a, \mathbf{b}, \mathbf{p})$ model, they follow a Dirichlet distribution with parameters $\mathbf{p} = (p_1, ..., p_D)$. The marginal distribution of part

```
## Quality of fit
par(mfrow=c(3,2))
hzbeta(uc,obj1)
mtext("Marginal distribution of the z-transform of parts, model obj1",
         line=-1,outer=TRUE)
```

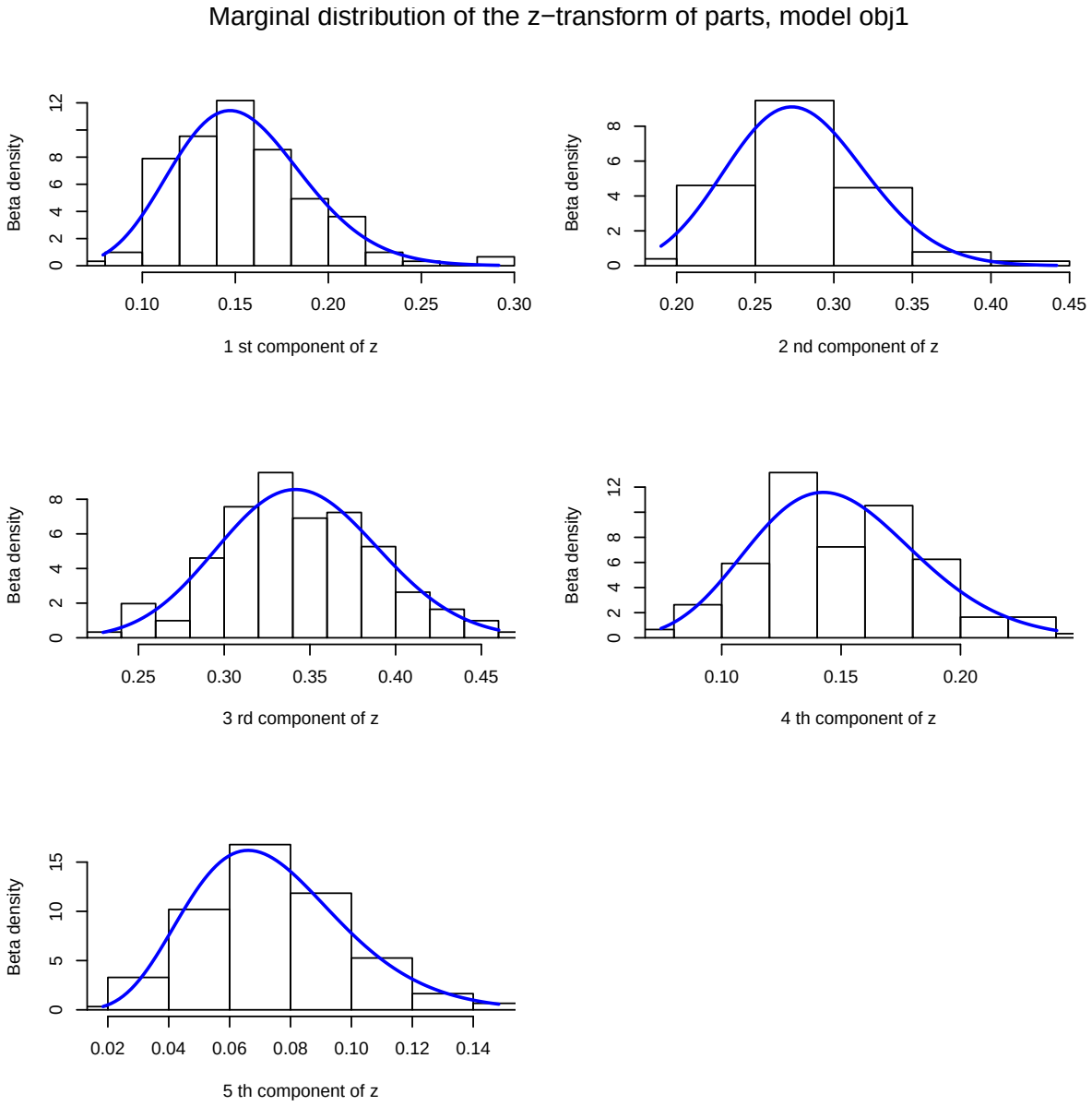


Figure 1: Histograms and marginal fitted Beta densities for the $z$-transforms of parts.

$z_i, i = 1, .., D$ is then $Beta(p_i)$. In Figure 1, histograms of the z-transforms of the compositions `uc` computed with the parameters estimated in `obj1` are compared with the beta densities. Notice that the ordinary expectation of $\mathbf{z}_j$ is $p_j / \sum_{i=1}^{D} p_i$, and the Aitchison expectation is $\exp[\psi(p_j)] / \sum_{i=1}^{D} \exp[\psi(p_i)]$.

## 4.2 Regression with isometric log-ratio transform

It is of interest to compare the preceding fit `obj1` with an similar model based on another log-ratio transform (lrt). Let us choose the matrix $\mathbf{V}$ in Equation (5) in such a way that the columns are orthonormal. The lrt so defined is an isometric log-ratio transform (ilr) (Egozcue et al., 2003).

```
## Influence of log-ratio transformation
# setup a matrix Vilr with orthonormal columns
Vilr <- contr.helmert(5)
Vilr <- Vilr%*%diag(1/sqrt(diag(crossprod(Vilr))))
colnames(Vilr) <- c("A.B","AB.C","A_C.D","A_D.E")
rownames(Vilr) <- colnames(uc)
print(Vilr)
```

```
##           A.B       AB.C      A_C.D      A_D.E
## SA -0.7071068 -0.4082483 -0.2886751 -0.2236068
## SB  0.7071068 -0.4082483 -0.2886751 -0.2236068
## SC  0.0000000  0.8164966 -0.2886751 -0.2236068
## SD  0.0000000  0.0000000  0.8660254 -0.2236068
## SE  0.0000000  0.0000000  0.0000000  0.8944272
```

The matrix `Vilr` is such a $\mathbf{V}$ matrix. It defines orthogonal contrasts in log of parts (Equation 5).

The formula implies the same form of dependence on explanatory variables as before.

```
Form2 <- Formula(A.B | AB.C | A_C.D | A_D.E ~  log(expend) +
        I(PAC*log(expend)) + log(sent) + log(FBCF) + log(price) + rates)
obj2 <-  regSGB(Form2, data = list(carseg, uc, Vilr), shape10 = 4.4,
                control.outer = list(trace=FALSE))
t2 <- table.regSGB(obj2)
## Comparison
# overall statistics
cbind(t1,t2)
```

```
##                   statistics   statistics
## value            401.4176523  401.4238180
## n.par             34.0000000   34.0000000
## n.par.fixed        0.0000000    0.0000000
## AIC             -734.8353045 -734.8476360
## Rsquare            0.8649558    0.8648061
## convergence        0.0000000    0.0000000
## kkt1               1.0000000    1.0000000
## kkt2               1.0000000    1.0000000
## counts.function  376.0000000  427.0000000
## counts.gradient  102.0000000  105.0000000
```

The overall statistics show the same log-likelihood `value` and `AIC`. The ilr fit `obj2` needed more iterations than `obj1`.

```
# fitted parts
range(obj1[["meanA"]]/obj2[["meanA"]] )
```

```
## [1] 0.9992912 1.0005631
```

The ratio of the fitted parts are very close to 1. Globally, the results do not depend on the chosen lrt (but see Section 4.4).

## 4.3 Different models for the lrt components

The model specified in the formula is the same for all lrt components. It is possible to specialize the model by fixing some of the parameters. For instance, the twelfth parameter in `obj1` is close to zero. It can be fixed to 0 by specifying the corresponding equality constraint with `heq=heqb.SGB, heq.jac=heqb.SGB.jac, ind=12` in the call to `regSGB`, see `EqualityConstr` in the package documentation for other examples.

## 4.4 Stepwise regression

`stepSGB` is an attempt to automatize the fixing of parameters in specific models for the lrt components. Starting with a fitted SGB regression, e.g. `obj1`, `stepSGB` fixes the regression parameters to 0, one parameter at a time in the decreasing order of the p-values in the starting model. There is the possibility to also fix `shape1` to some given value $a$. The Dirichlet parameters `shape2` cannot be fixed. At each iteration, the model is fitted and the AIC criterion computed. The procedure stops when the AIC is increasing.

```
## stepSGB
# First lrt
step1 <- stepSGB(obj1, carseg, uc, bound = 2.1, control.outer = list(trace=FALSE))


## [1] AIC = -734.835304506414
## [1] indices of fixed parameters:
## [1] 12
## [1] AIC = -736.882066132937
## [1] indices of fixed parameters:
## [1] 12 19
## [1] AIC = -738.570726355319
## [1] indices of fixed parameters:
## [1] 12 19 25
## [1] AIC = -740.356645097694
## [1] indices of fixed parameters:
## [1] 12 19 25 16
## [1] AIC = -741.649925076279
## [1] indices of fixed parameters:
## [1] 12 19 25 16 27
## [1] AIC = -731.910777863515


round(step1[["tab"]])


##                 full iter1 iter2 iter3 iter4 iter5
## value            401   401   401   401   401   395
## n.par             34    34    34    34    34    34
## n.par.fixed        0     1     2     3     4     5
## AIC             -735  -737  -739  -740  -742  -732
## Rsquare            1     1     1     1     1     1
## convergence        0     0     0     0     0     0
## kkt1               1     1     1     1     1     1
## kkt2               1     1     1     1     1     1
```

```
## counts.function  376  1085  1532  1735  1837  2005
## counts.gradient  102   197   271   282   311   435
```

Starting with full model `obj1`, the procedure stopped after 5 iterations. The best model (minimum AIC) is given at the second last iteration.

A summary gives the results at iteration 4. The standard errors are conditional to the fixed parameters.

```
summary(step1[["reg"]][["iter4"]])


## $call
## regSGB.default(d = d, u = as.matrix(u), V = obj0[["V"]], weight = weight,
##     shape10 = shape10, bound = bound, ind = indi, shape1 = shape1,
##     Mean2 = Mean2, control.optim = control.optim, control.outer = control.outer)
##
## $Formula
## AB | BC | CD | DE ~ log(expend) + I(PAC * log(expend)) + log(sent) +
##     log(FBCF) + log(price) + rates
##
## $parameters
##                 Parameters Initial Estimate StdErr1 StdErr p.value signif
## 1                   shape1   2.359    2.154   0.069  0.076   0.000    ***
## 2           (Intercept).AB  -1.464   -2.215   0.644  0.832   0.008     **
## 3           (Intercept).BC -10.512  -10.606   1.385  1.795   0.000    ***
## 4           (Intercept).CD -23.778  -22.572   1.184  1.555   0.000    ***
## 5           (Intercept).DE -20.484  -24.586   0.527  0.685   0.000    ***
## 6           log(expend).AB   0.706    0.745   0.278  0.279   0.008     **
## 7           log(expend).BC   0.638    0.807   0.122  0.153   0.000    ***
## 8           log(expend).CD   2.780    2.634   0.262  0.266   0.000    ***
## 9           log(expend).DE   3.187    3.482   0.280  0.279   0.000    ***
## 10 I(PAC * log(expend)).AB   0.047    0.047   0.003  0.003   0.000    ***
## 11 I(PAC * log(expend)).BC   0.013    0.014   0.002  0.002   0.000    ***
## 12 I(PAC * log(expend)).CD   0.000    0.000   0.000  0.000      NA  fixed
## 13 I(PAC * log(expend)).DE   0.012    0.012   0.004  0.003   0.000    ***
## 14            log(sent).AB  -0.641   -0.613   0.137  0.131   0.000    ***
## 15            log(sent).BC   0.296    0.343   0.101  0.095   0.000    ***
## 16            log(sent).CD   0.124    0.000   0.000  0.000      NA  fixed
## 17            log(sent).DE  -0.679   -0.611   0.158  0.144   0.000    ***
## 18            log(FBCF).AB  -0.603   -0.552   0.364  0.348   0.113
## 19            log(FBCF).BC   0.216    0.000   0.000  0.000      NA  fixed
## 20            log(FBCF).CD  -0.789   -0.721   0.308  0.313   0.021      *
## 21            log(FBCF).DE  -1.290   -1.278   0.337  0.332   0.000    ***
## 22           log(price).AB   0.531    0.507   0.116  0.123   0.000    ***
## 23           log(price).BC  -0.382   -0.341   0.076  0.093   0.000    ***
## 24           log(price).CD   0.229    0.263   0.095  0.102   0.010      *
## 25           log(price).DE   0.139    0.000   0.000  0.000      NA  fixed
## 26                rates.AB   0.101    0.100   0.017  0.017   0.000    ***
## 27                rates.BC   0.017    0.025   0.007  0.006   0.000    ***
## 28                rates.CD  -0.034   -0.034   0.015  0.014   0.016      *
## 29                rates.DE   0.043    0.043   0.017  0.015   0.004     **
## 30               shape2.SA  17.703   19.204   1.764  1.790   0.000    ***
## 31               shape2.SB  17.703   34.506   0.101  0.105   0.000    ***
## 32               shape2.SC  17.703   43.095   0.130  0.130   0.000    ***
## 33               shape2.SD  17.703   18.483   1.914  1.967   0.000    ***
```

```
## 34               shape2.SE  17.703    9.205   1.221  1.268   0.000    ***
##
## $signif.codes
## [1] " 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1"
##
## attr(,"class")
## [1] "summary.regSGB"
```

```
# Second lrt
step2 <- stepSGB(obj2, carseg, uc, bound = 2.1, control.outer = list(trace=FALSE))

## [1] AIC = -734.847636007667
## [1] indices of fixed parameters:
## [1] 15
## [1] AIC = -736.872648398603
## [1] indices of fixed parameters:
## [1] 15 19
## [1] AIC = -738.595606724564
## [1] indices of fixed parameters:
## [1] 15 19  2
## [1] AIC = -740.294040872104
## [1] indices of fixed parameters:
## [1] 15 19  2 23
## [1] AIC = -740.879289413642
## [1] indices of fixed parameters:
## [1] 15 19  2 23 24
## [1] AIC = -741.780290882074
## [1] indices of fixed parameters:
## [1] 15 19  2 23 24 16
## [1] AIC = -743.20687516917
## [1] indices of fixed parameters:
## [1] 15 19  2 23 24 16 28
## [1] AIC = -745.101440642451
## [1] indices of fixed parameters:
## [1] 15 19  2 23 24 16 28 18
## [1] AIC = -745.093474402558

round(step2[["tab"]])

##                full iter1 iter2 iter3 iter4 iter5 iter6 iter7 iter8
## value           401   401   401   401   400   400   400   400   399
## n.par            34    34    34    34    34    34    34    34    34
## n.par.fixed       0     1     2     3     4     5     6     7     8
## AIC            -735  -737  -739  -740  -741  -742  -743  -745  -745
## Rsquare           1     1     1     1     1     1     1     1     1
## convergence       0     0     0     0     0     0     0     0     0
## kkt1              1     1     1     1     1     1     1     1     1
## kkt2              1     1     1     1     1     1     1     1     1
## counts.function 427  1246  1236  1409  1647  1831  1742  2375  2415
## counts.gradient 105   216   228   291   319   331   324   415   417
```

If we call `stepSGB` with full model `obj2` (with the ilr transforms), we see that 7 parameters can be fixed and that the AIC is slightly smaller than the best AIC in `step1`. The fixed parameters involve essentially

AB.C and A_C.D.

```
summary(step2[["reg"]][["iter7"]])
```

```
## $call
## regSGB.default(d = d, u = as.matrix(u), V = obj0[["V"]], weight = weight,
##     shape10 = shape10, bound = bound, ind = indi, shape1 = shape1,
##     Mean2 = Mean2, control.optim = control.optim, control.outer = control.outer)
##
## $Formula
## A.B | AB.C | A_C.D | A_D.E ~ log(expend) + I(PAC * log(expend)) +
##     log(sent) + log(FBCF) + log(price) + rates
##
## $parameters
##                     Parameters Initial Estimate StdErr1 StdErr p.value signif
## 1                       shape1   2.326    1.878   0.060  0.063   0.000    ***
## 2             (Intercept).A.B   1.035    0.000   0.000  0.000      NA  fixed
## 3            (Intercept).AB.C   9.180    6.800   0.931  1.007   0.000    ***
## 4           (Intercept).A_C.D  27.084   28.390   1.097  1.178   0.000    ***
## 5           (Intercept).A_D.E  39.300   40.565   0.031  0.033   0.000    ***
## 6             log(expend).A.B  -0.499   -0.405   0.212  0.192   0.035      *
## 7            log(expend).AB.C  -0.809   -0.554   0.082  0.088   0.000    ***
## 8           log(expend).A_C.D  -2.980   -2.893   0.141  0.139   0.000    ***
## 9           log(expend).A_D.E  -5.158   -5.299   0.317  0.314   0.000    ***
## 10   I(PAC * log(expend)).A.B  -0.033   -0.034   0.002  0.002   0.000    ***
## 11  I(PAC * log(expend)).AB.C  -0.030   -0.031   0.002  0.002   0.000    ***
## 12 I(PAC * log(expend)).A_C.D  -0.021   -0.019   0.002  0.002   0.000    ***
## 13 I(PAC * log(expend)).A_D.E  -0.027   -0.029   0.004  0.003   0.000    ***
## 14              log(sent).A.B   0.453    0.446   0.096  0.083   0.000    ***
## 15             log(sent).AB.C   0.020    0.000   0.000  0.000      NA  fixed
## 16            log(sent).A_C.D  -0.093    0.000   0.000  0.000      NA  fixed
## 17            log(sent).A_D.E   0.536    0.528   0.152  0.129   0.000    ***
## 18              log(FBCF).A.B   0.427    0.401   0.282  0.250   0.109
## 19             log(FBCF).AB.C   0.070    0.000   0.000  0.000      NA  fixed
## 20            log(FBCF).A_C.D   0.733    0.456   0.124  0.130   0.000    ***
## 21            log(FBCF).A_D.E   1.721    1.821   0.425  0.411   0.000    ***
## 22             log(price).A.B  -0.376   -0.419   0.084  0.080   0.000    ***
## 23            log(price).AB.C   0.095    0.000   0.000  0.000      NA  fixed
## 24           log(price).A_C.D  -0.131    0.000   0.000  0.000      NA  fixed
## 25           log(price).A_D.E  -0.226   -0.237   0.131  0.115   0.039      *
## 26                 rates.A.B  -0.072   -0.070   0.013  0.013   0.000    ***
## 27                rates.AB.C  -0.055   -0.050   0.005  0.005   0.000    ***
## 28               rates.A_C.D  -0.010    0.000   0.000  0.000      NA  fixed
## 29               rates.A_D.E  -0.046   -0.052   0.020  0.018   0.004     **
## 30                 shape2.SA  18.195   25.052   0.198  0.203   0.000    ***
## 31                 shape2.SB  18.195   45.629   0.076  0.077   0.000    ***
## 32                 shape2.SC  18.195   55.821   0.060  0.064   0.000    ***
## 33                 shape2.SD  18.195   24.019   0.280  0.293   0.000    ***
## 34                 shape2.SE  18.195   11.978   1.590  1.646   0.000    ***
##
## $signif.codes
## [1] " 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1"
##
## attr(,"class")
## [1] "summary.regSGB"
```

## 4.5 Comparison of fits

Two goodness of fit tests are proposed in the R-package **SGB**: Kolmogorov-Smirnov and Cramer-von-Mises. They test the adequation between the Beta distribution with the fitted parameters and the $z(u)$-transforms of parts. The four fits `obj1`, `obj2`, `st14 = step1[["reg"]][["iter4"]]` and `st27 = step2[["reg"]][["iter7"]]` are compared below with Kolmogorov-Smirnov.

```
## gof  tests
npar <- length(obj1[["par"]])
D <- dim(uc)[2]
np2 <- npar-D+1

# K-S test on obj1
ks1 <- ks.SGB(uc, shape1=obj1[["par"]][1], shape2=obj1[["par"]][np2:npar],
              scale=obj1[["scale"]])[["tests"]][,1:2]
names(ks1) <- c("stat1","pval1")

# K-S test on obj2
ks2 <- ks.SGB(uc, shape1=obj2[["par"]][1], shape2=obj2[["par"]][np2:npar],
              scale=obj2[["scale"]])[["tests"]][,1:2]
names(ks2) <- c("stat2","pval2")

st14 <- step1[["reg"]][["iter4"]]
st27 <- step2[["reg"]][["iter7"]]

# K-S test on st14
ks14 <- ks.SGB(uc, shape1=st14[["par"]][1], shape2=st14[["par"]][np2:npar],
               scale=st14[["scale"]])[["tests"]][,1:2]
names(ks14) <- c("stat14","pval14")

# K-S test on st2
ks27 <- ks.SGB(uc, shape1=st27[["par"]][1], shape2=st27[["par"]][np2:npar],
               scale=st27[["scale"]])[["tests"]][,1:2]
names(ks27) <- c("stat27","pval27")

## Kolmogorov-Smirnov tests
round(cbind(ks1,ks2,ks14,ks27),3)
```

```
##   stat1 pval1 stat2 pval2 stat14 pval14 stat27 pval27
## 5 0.060 0.640 0.061 0.629  0.060  0.636  0.063  0.591
## 4 0.055 0.741 0.055 0.741  0.055  0.755  0.054  0.761
## 1 0.041 0.960 0.042 0.953  0.043  0.946  0.054  0.774
## 3 0.039 0.973 0.040 0.969  0.041  0.959  0.050  0.846
## 2 0.039 0.977 0.039 0.976  0.033  0.997  0.042  0.954
```

The four fits perform similarly. For each part, the Beta-distribution is accepted (but remember that the tests are conservative).

## 4.6 Imputation of missing parts

Applied to a completely missing composition, `impute.regSGB` returns the Aitchison expectation (Equation 3). Applied to a partially missing composition, it returns the conditional Aitchison expectation, given the observed sub-composition (Equation 4). Applied to a complete case, it returns the complete case.

```
usup <- carseg[1:3,1:5]
## Introduce some missing values
usup[1,2] <- NA
usup[2,2:3] <- NA
usup[3,] <- NA
usup <- usup/rowSums(usup,na.rm=TRUE)
usup
```

```
##          SA SB        SC        SD        SE
## 1 0.1103144 NA 0.5046769 0.2387415 0.1462672
## 2 0.2348730 NA        NA 0.5004687 0.2646583
## 3        NA NA        NA        NA        NA
```

```
impute.regSGB(obj1,carseg[1:3,],usup)
```

```
##           SA        SB        SC        SD         SE
## 1 0.07133878 0.3533139 0.3263676 0.1543908 0.09458895
## 2 0.07068115 0.3299802 0.3690864 0.1506078 0.07964454
## 3 0.06788780 0.3411772 0.3569344 0.1604960 0.07350466
```

```
## original values
carseg[1:3,1:5]
```

```
##           SA        SB        SC        SD         SE
## 1 0.06927409 0.3720303 0.3169218 0.1499224 0.09185135
## 2 0.07309030 0.3295972 0.3592120 0.1557412 0.08235921
## 3 0.06434496 0.3051185 0.3579021 0.1855172 0.08711717
```

The reconstructed parts are close to the original ones.

# 5 Summary

In this paper, the main features of the R-package **SGB** are explained. Regression models based on a generalization of the Dirichlet distribution, called the Simplicial Beta distribution (SGB), are defined and exemplified. Under the SGB, other parameters are added to the Dirichlet shape parameters: an overall shape parameter and a scale composition. Log-ratio transforms of the scale compositions are modeled with auxiliary variables. The shape parameters are the same across compositions. This is in contrast with Dirichlet regressions, where the Dirichlet shapes are the only parameters and are modeled with auxiliary variables. The fitted compositions are given by the Aitchison expectation, i.e. the image in the simplex of the expected log-ratios. These Aitchison expectations depend on all parameters, the shape parameters modifying the constant term in the regressions (see Equation 3). The results in the simplex do not depend on the chosen log-ratio transforms (lrt). Nevertheless the importance of the choice of the lrt appears when different models are specified for its components. This has been exemplified with backward stepwise regressions. The principle is to define first the same overall model for all lrt components and then to set specific regression parameters to zero. It is also possible to fix the overall shape to a given value. The regression parameters are set to zero in decreasing order of their asymptotic p-value. The elimination stops when the AIC criterion increases. Notice that the initial p-values are used throughout. The standard deviations of the final model are those of the distribution conditional on the fixed values. SGB regression proves to be quite feasible and flexible. It sheds a new light on the modeling of compositions.

# References


Aitchison, J. (1986). *The Statistical Analysis of Compositional Data*. Monographs on Statistics and Applied Probability. Chapman and Hall Ltd (reprinted 2003 with additional material by the Blackburn Press, London (UK).

Craiu, M. and V. Craiu (1969). Repartitia Dirichlet generalizatá. *Analele Universitatii Bucuresti, Mathematicá-Mecanicá 18*, 9–11.

Egozcue, J. J. and V. Pawlovsky-Glahn (2011). Chapter 2: Basic concepts and procedures. In V. Pawlowsky-Glahn and A. Buccianti (Eds.), *Compositional data analysis. Theory and applications*. Wiley.

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

Gilbert, P. and R. Varadhan (2016). *numDeriv: Accurate Numerical Derivatives*. R package version 2016.8-1.

Graf, M. (2011). Chapter 9: Use of survey weights for the analysis of compositional data. In V. Pawlowsky-Glahn and A. Buccianti (Eds.), *Compositional data analysis. Theory and applications*. Wiley.

Graf, M. (2017). A distribution on the simplex of the Generalized Beta type. In J. A. Martín-Fernández (Ed.), *Proceedings CoDaWork 2017*. University of Girona (Spain).

Grün, B., I. Kosmidis, and A. Zeileis (2012). Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned. *Journal of Statistical Software 48*(11), 1–25.

Gueorguieva, R., R. Rosenheck, and D. Zelterman (2008). Dirichlet Component Regression and its Applications to Psychiatric Data. *Comput Stat Data Anal. 52*(12), 5344–5355.

Hijazi, R. H. and R. W. Jernigan (2009). Modelling compositional data using Dirichlet regression models. *Journal of Applied Probability & Statistics, 4*(1), 77–91.

Huber, P. J. (1967). The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, pp. 221–233.

Kotz, S., N. Balakrishnan, and N. L. Johnson (2000). *Continuous Multivariate Distributions, Volume 1, Models and Applications*. John Wiley & Sons.

Madsen, K., H. Nielsen, and O. Tingleff (2004). Optimization With Constraints. Informatics and Mathematical Modelling, Technical University of Denmark.

Maier, M. J. (2015). *DirichletReg: Dirichlet Regression in R*. R package version 0.6-3.1.

Monti, G. S., G. Mateu-Figueras, and V. Pawlowsky-Glahn (2011). Notes on the scaled Dirichlet distribution. In V. Pawlowsky-Glahn and A. Buccianti (Eds.), *Compositional data analysis. Theory and applications*. Wiley.

Morais, J. and C. Thomas-Agnan (2019). Impact of economic context on automobile market segment shares: a compositional approach. In press.

Ng, K. W., G.-L. Tian, and M.-L. Tang (2011). *Dirichlet and related distributions: theory, methods and applications*. Wiley series in probability and statistics.

Templ, M., K. Hron, and P. Filzmoser (2011). *robCompositions: an R-package for robust statistical analysis of compositional data*. John Wiley and Sons.

van den Boogaart, K. G., R. Tolosana, and M. Bren (2014). *compositions: Compositional Data Analysis*. R package version 1.40-1.

Varadhan, R. (2015). *alabama: Constrained Nonlinear Optimization*. R package version 2015.3-1.

Wicker, N., J. Muller, R. K. R. Kalathur, and O. Poch (2008). A maximum likelihood approximation method for Dirichlet's parameter estimation. *Computational Statistics & Data Analysis 52*(3), 1315–1322.

Zeileis, A. and Y. Croissant (2010). Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software 34*(1), 1–13.