

Package ‘DirichletReg’

January 27, 2015

Type Package

Version 0.6-1

Date 2014-12-12

Title Dirichlet Regression in R

Encoding UTF-8

Depends R (>= 3.0.0), Formula, rgl

Imports stats, graphics, methods, maxLik

Suggests testthat

Description Implements the Dirichlet Regression in R

License GPL (>= 2)

URL <http://dirichletreg.r-forge.r-project.org/>
<http://r-forge.r-project.org/projects/dirichletreg/>
<http://CRAN.R-project.org/package=DirichletReg>

ByteCompile yes

LazyLoad yes

LazyData yes

ZipData yes

Author Marco Johannes Maier [cre, aut]

Maintainer Marco Johannes Maier <marco.maier@wu.ac.at>

Repository CRAN

Repository/R-Forge/Project dirichletreg

Repository/R-Forge/Revision 25

Repository/R-Forge/DateTimeStamp 2014-12-08 18:28:11

Date/Publication 2014-12-10 18:50:30

NeedsCompilation yes

R topics documented:

DirichletReg-package	2
anova.DirichletRegModel	2
ArcticLake	3
BloodSamples	4
Dirichlet	5
DirichletRegData	6
DirichletRegModel	9
DirichReg	11
GlacialTills	14
plot.DirichletRegData	15
Reading Accuracy Data	16
Rocks	16
Simplex-Transformations	17

Index	19
--------------	-----------

DirichletReg-package *The **DirichletReg** Package*

Description

This package provides a functions to analyze compositional data using Dirichlet regression methods.

Package: DirichletReg
 Type: Package
 Version: 0.6-1
 Date: 2014-12-12
 License: GPL (>= 2)

Author(s)

Marco J. Maier

Examples

```
example(plot.DirichletRegData)
example(DirichReg)
```

anova.DirichletRegModel

Compare Dirichlet Regression Models using an LRT

Description

This function allows for pairwise tests of Dirichlet regression models using a likelihood ratio test (LRT).

Usage

```
## S3 method for class 'DirichletRegModel'
anova(object, ..., sorted = FALSE)
```

Arguments

object	the model to be compared against those listed in ...
...	models to be tested against the one specified as object
sorted	should the models be sorted according to their numbers or parameters?

Details

The test statistic is computed $LR = -2 [\log(L_a) - \log(L_b)]$ where L_i is the likelihood of model i with df equal to the difference of the number of parameters in the models.

Author(s)

Marco J. Maier

Examples

```
ALake <- ArcticLake
ALake$AL <- DR_data(ArcticLake[,1:3])
mod0 <- DirichReg(AL ~ 1, ALake)
mod1 <- DirichReg(AL ~ depth, ALake)
mod2 <- DirichReg(AL ~ depth + I(depth^2), ALake)
anova(mod1, mod0, mod2, sorted = TRUE)
```

ArcticLake

Arctic Lake Data (Aitchison)

Description

These data are taken from Aitchison (2003) and contain information on the relation of sediment composition with depth in an Arctic lake.

Usage

```
ArcticLake
```

Format

A data frame with 39 observations on the following 4 variables:

sand, silt, clay relative frequencies of sand, silt, and clay
 depth water depth in meters

Source

Aitchison, J. (2003). *The Statistical Analysis of Compositional Data*. The Blackburn Press, Caldwell, NJ.

Examples

```
head(ArcticLake)
AL <- DR_data(ArcticLake[,1:3])
plot(AL)
summary(AL)
```

 BloodSamples

Serum Protein Composition in Blood Samples

Description

These data (Aitchison, 2003) list blood samples' compositions of *Albumin*, *Pre-Albumin*, *Globulin A*, and *Globulin B* in relation to two types of diseases. 14 patients suffer from disease A, 16 from disease B and 6 are unclassified.

Usage

BloodSamples

Format

A data frame with 36 observations on the following 6 variables.

Albumin, Pre.Albumin, Globulin.A, Globulin.B the amounts of Albumin, Pre-Albumin, Globulin A, and Globulin B.

Disease diagnosis of disease A, B, or NA for unclassified observations.

New a factor indicating whether the observations are old and classified (No) or new and unclassified (Yes).

Source

Aitchison, J. (2003). *The Statistical Analysis of Compositional Data*. The Blackburn Press, Caldwell, NJ.

Examples

```
head(BloodSamples)
Bl <- DR_data(BloodSamples[,1:4])
summary(Bl)
```

Dirichlet

*The Dirichlet Distribution***Description**

Density function and random number generation for the Dirichlet distribution

Usage

```
rdirichlet(n, alpha)

ddirichlet(x, alpha, log = FALSE, sum.up = FALSE)

ddirichlet_R(x, alpha, log = FALSE, sum.up = FALSE)
```

Arguments

n	number of random observations to draw
x	a matrix containing observations
alpha	the Dirichlet distribution's parameters. Can be a vector (one set of parameters for all observations) or a matrix (a different set of parameters for each observation), see "Details"
log	if TRUE, logarithmic densities are returned
sum.up	if TRUE, the (log-)likelihood is returned

Details

The Dirichlet distribution is a multidimensional generalization of the Beta distribution where each dimension is governed by an α -parameter. Formally this is

$$\mathcal{D}(\alpha_i) = \left[\Gamma\left(\sum_i \alpha_i\right) / \prod_i \Gamma(\alpha_i) \right] \prod_i y_i^{\alpha_i - 1}$$

Usually, alpha is a vector thus the same parameters will be used for all observations. If alpha is a matrix, a complete set of α -parameters must be supplied for each observation.

log returns the logarithm of the densities (therefore the log-likelihood) and sum.up returns the product or sum and thereby the likelihood or log-likelihood.

Dirichlet (log-)densities are by default computed using C-routines (ddirichlet_log_vector and ddirichlet_log_matrix), a version only using R is provided by ddirichlet_R. Caution: Although .C() can be used to call the C routines directly, R will crash or produce wrong values, if, e.g., data types are not set properly.

Value

<code>rdirichlet</code>	returns a matrix with random numbers according to the supplied alpha vector or matrix.
<code>ddirichlet</code>	returns a vector of densities (if <code>sum.up = FALSE</code>) or the (log-)likelihood (if <code>sum.up = TRUE</code>) for the given data and alphas.
<code>ddirichlet_R</code>	as <code>ddirichlet</code> , only implemented purely in R.

Author(s)

Marco J. Maier

Examples

```
X1 <- rdirichlet(100, c(5, 5, 10))

a.mat <- cbind(1:10, 5, 10:1)
a.mat
X2 <- rdirichlet(10, a.mat)
# note how the probabilities in the first an last column relate to a.mat
round(X2, 2)

ddirichlet(X1, c(5, 5, 10))
ddirichlet(X2, a.mat)
```

DirichletRegData *Prepare Compositional Data*

Description

This function prepares a matrix with compositional variables for further processing in the **DirichletReg** package.

Usage

```
DR_data(Y, trafo = sqrt(.Machine$double.eps), base = 1,
        norm_tol = sqrt(.Machine$double.eps))

## S3 method for class 'DirichletRegData'
print(x, type = c("processed", "original"), ...)

## S3 method for class 'DirichletRegData'
summary(object, ...)
```

Arguments

Y	A matrix or data.frame with nonnegative values of all compositional variables (in some cases, a vector is also permissible, see “Details”).
trafo	Either a logical or numeric value. Transformation of variables causes the values to shrink away from extreme values of 0 and 1, see “Details”. If logical, it will force (TRUE) or suppress (FALSE) transformation. Suppressing transformation in the presence of extreme values (0 and 1) will result in an error. If trafo is numeric it is used as a “threshold”, so transformation will be applied if values in Y are $y < \text{trafo}$ or $y > (1 - \text{trafo})$.
base	The “base” component to use in the reparametrized model
norm_tol	Due to numerical precision, row sums of Y may not be <i>exactly</i> equal to 1. Therefore, norm_tol is a small non-negative value (default: $\sqrt{\text{Machine}\$double.eps}$) which represents the tolerance when testing for “near equality” to 1 (see all.equal).
x	A DirichletRegData object
type	Displays either the (possibly normalized or transformed) “processed” or “original” data
object	A DirichletRegData object
...	Further arguments

Details**Y:**

Y is a matrix or data.frame containing compositional variables. If they do not sum up to 1 for all observations, normalization is forced where each row entry is divided by the row’s sum (a warning will be issued that normalization was applied).

In case one row-entry (or more) is NA, the whole row will be returned as NA. Beta-distributed variables can be supplied as a single vector which, however, has to have values in the interval [0, 1]. The second variable will be generated (1 - Y) and a matrix consisting of the columns 1 - Y and Y will be returned. A message will be issued that a beta-distributed variable was assumed and that this assumption needs to be checked.

trafo:

The transformation (done if trafo = TRUE) is a generalization of that proposed by Smithson and Verkuilen (2006) that transforms each component y of Y by computing $y^* = \frac{y(n-1) + \frac{1}{2}}{n}$ where n is the number of observations in Y (this approach is also used in the package **betareg**, see Cribari-Neto & Zeileis, 2010).

For an arbitrary number of dimensions (or variables) d the transformation is $y^* = \frac{y(n-1) + \frac{1}{d}}{n}$.

base:

To set the base (i.e., omitted) component of Y for the “alternative” (mean/precision) model, the argument base can be used. This is by default set to the first variable in Y (if a vector is supplied, the column 1 - Y becomes the base component).

Note that the definition can be overruled in [DirichReg](#).

x and object:

Objects created by DR_data.

type:

specifies for the print method whether the original or processed data are displayed.

Value

The function returns a matrix object of class `DirichletRegData` with the following attributes:

```
attr(*, "dimnames")
    a list with two entries, row names (by default NULL) and column names.
attr(*, "Y.original")
    the original data
attr(*, "dims")
    number of dimensions of Y (i.e., number of columns)
attr(*, "dim.names")
    the number of components in Y
attr(*, "obs")
    number of observations of Y (i.e., number of rows)
attr(*, "valid_obs")
    number of valid observations
attr(*, "normalized")
    a logical value indicating whether the data were normalized
attr(*, "transformed")
    a logical value indicating whether the data were transformed
attr(*, "base")
    number of the variable used as the base in the reparametrized model
```

Author(s)

Marco J. Maier

References

Smithson, M. & Verkuilen, J. (2006). A Better Lemon Squeezer? Maximum-Likelihood Regression With Beta-Distributed Dependent Variables. *Psychological Methods*, 11(1), 54–71.

Cribari-Neto, F. & Zeileis, A. (2010). Beta Regression in R. *Journal of Statistical Software*, 34(2), 1–24.

Examples

```
# create a DirichletRegData object from the Arctic Lake data
head(ArcticLake[, 1:3])
AL <- DR_data(ArcticLake[, 1:3])
summary(AL)
head(AL)
```

 DirichletRegModel *Methods for the Class DirichletRegModel*

Description

These are available methods for the results of Dirichlet regression models and objects of class `DirichletRegModel`. These methods contain functions for print and summary of the data, generate fitted values and predicting new values using `predict`. Various types of residuals are implemented and `confint` can be used to compute confidence intervals of the parameters. Furthermore `logLik` extracts the log-likelihood of the model and `vcov` extracts the covariance matrix of the parameter estimates.

Usage

```
## S3 method for class 'DirichletRegModel'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'DirichletRegModel'
summary(object, ...)

## S3 method for class 'DirichletRegModel'
fitted(object, mu = TRUE, alpha = FALSE, phi = FALSE, ...)

## S3 method for class 'DirichletRegModel'
predict(object, newdata, mu = TRUE, alpha = FALSE, phi = FALSE, ...)

## S3 method for class 'DirichletRegModel'
residuals(object, type = c("standardized", "composite", "raw"), ...)

## S3 method for class 'DirichletRegModel'
confint(object, parm, level, ..., type=c("all", "beta", "gamma"), exp = FALSE)

## S3 method for class 'DirichletRegConfint'
print(x, digits = 3, ...)

## S3 method for class 'DirichletRegModel'
logLik(object, ...)

## S3 method for class 'DirichletRegModel'
AIC(object, ..., k = 2)

## S3 method for class 'DirichletRegModel'
BIC(object, ...)

## S3 method for class 'DirichletRegModel'
nobs(object, ...)
```

```
## S3 method for class 'DirichletRegModel'
vcov(object, ...)

## S3 method for class 'DirichletRegModel'
update(object, formula., ..., evaluate = TRUE)

## S3 method for class 'DirichletRegModel'
drop1(object, scope, test = c("LRT", "none"), k = 2, ...)
```

Arguments

x	an object of class <code>DirichletRegModel</code>
object	an object of class <code>DirichletRegModel</code> or <code>DirichletRegConfint</code> for printing an object obtained by <code>confint.DirichletRegModel</code>
alpha	logical; returns alpha values
mu	logical; returns expected values
phi	logical; returns precision values
type	for residuals: defines the type of residuals to be computed "standardized" (i.e., Pearson), "composite", or "raw" for <code>confint</code> : defines the type of parameter ("all", "beta", or "gamma") for which confidence values are returned
newdata	a <code>data.frame</code> containing new observations
k	number for the weighting of parameters
parm	a vector containing names of the parameters to print
level	(a vector of) confidence level(s), defaults to .95
exp	logical; returns parameters in exponentiated form
digits	the number of digits in the output
formula.	the new formula to be updated, see update.Formula
evaluate	if FALSE the updated call will be returned, but not evaluated
scope	defines the scope of variables to be dropped, see drop1
test	defines the type of test for <code>drop1</code>
...	further arguments

Author(s)

Marco J. Maier

Examples

```
ALake <- ArcticLake
ALake$AL <- DR_data(ArcticLake[, 1:3])

mod1 <- DirichReg(AL ~ depth + I(depth^2) | depth, data = ALake, model="alternative")

update(mod1, . ~ . | . + I(depth^2), evaluate = FALSE)
```

```

mod1

drop1(mod1)  ### issues a caveat when used for the first time in an R session

summary(mod1)

head(fitted(mod1))

predict(mod1, newdata = data.frame("depth" = seq(10, 100, 10)))

head(residuals(mod1))

confint(mod1)
confint(mod1, exp = TRUE)

logLik(mod1)
round(vcov(mod1), 5)

```

DirichReg

Fitting a Dirichlet Regression

Description

This function allows for fitting Dirichlet regression models using two different parametrizations.

Usage

```
DirichReg(formula, data, model = c("common", "alternative"),
          subset, sub.comp, base, weights, control, verbosity = 0)
```

Arguments

formula	the model formula (for different specifications see “Details”)
data	a data.frame containing independent and dependent variables
model	specifies whether the “common” (α s) or “alternative” (μ/ϕ) parametrization is employed (see “Details”)
subset	estimates the model for a subset of the data
sub.comp	analyze a subcomposition by selecting specific components (see “Details”)
base	redefine the base variable
weights	frequency weights
control	a list containing control parameters used for the optimization
verbosity	prints information about the function’s progress, see Details

Details

Formula Specification and Models: formula determines the used predictors. The responses **must** be prepared by `DR_data` and can be optionally stored in the object containing all covariates which is then specified as the argument `data`. (Although “on-the-fly” processing of `DR_data` in a formula works, it is only intended for testing purposes and may be removed at any time – use at your own risk.)

There are two different parametrization (controlled by the argument `model`, see below):

- the “*common*” param. that models each α by an (possibly individual) set of predictors, and
- the “*alternative*” param. that models expected values (μ ; as in multinomial logistic regression) and precision parameters (ϕ) with two sets of predictors.

As the two models offer different modeling strategies, the specification of their formulae differ:

Formulae for the “Common” Model: The simplest possible model here is to include only an intercept for all components. If DV is the ‘dependent variable’ (i.e., compositional data) with three components, we can request this null-model by $DV \sim 1$. We always have at least two dependent variables, so simple formulae as the one given above will be expanded to $DV \sim 1 | 1 | 1$, because DV has three components. Likewise, it is possible to specify a common set of predictors for all components, as in $DV \sim p1 * p2$, where `p1` and `p2` are predictors.

If the covariates of the components shall differ, one has to set up a complete formula for each subcomposition, using `|` as separators between the components, for example, $DV \sim p1 | p1 + p2 | p1 * p2$ will lead to a model where the first response in DV will be modeled using `p1`, the second will be predicted by `p1 + p2` and the third by `p1 * p2`. Note that if you use the latter approach, the predictors have to be stated explicitly for all response variables.

Formulae for the “Alternative” Model: The simplest possible model here is to include an intercept for all components (except the base) and an intercept for precision. This can be achieved by $DV \sim 1$, which is expanded to $DV \sim 1 | 1$. The part modeling the ‘mean’ (first element on the right-hand side) is mandatory, if no specification for precision is included, an intercept will be added. Note that you need to set `model = "alternative"` to use this parametrization!

The alternative parametrization consists of two parts: modeled expected values (μ) and their ‘precision’ (ϕ). As in multinomial logistic regression, one response variable is omitted (by default the first, but this can be changed by the base argument in `DR_data` or `DirichReg`) and for the rest a set of predictors is used with a multinomial logit-link. For precisions, a different set of predictors can be set up using a log-link.

$DV \sim p1 * p2 | p1 + p2$ will set up a model where the expected values are predicted by `p1 * p2` and precision are modeled using `p1 + p2`.

Data Preparation: The data argument accepts a `data.frame` that **must** include the dependent variable as a named element (see examples how to do this).

Changing the Base Component and Analyzing Subcompositions: The base-component (i.e., omitted component) is initially set during the stage of data preparation `DR_data`, but can easily be changed using the argument `base` which takes integer values from 1 to the maximum number of components.

If a data set contains a large number of components, of which only a few are relevant, the latter can be ‘sorted out’ and the irrelevant (i.e., not selected) components will be aggregated into a single variable (row sums) that automatically becomes the base category for the model, unless specified otherwise by `base`. The positioning of variables will necessarily change: the aggregated variable takes the first column and the others are appended in their order of selection.

Subsets and Weights: Using `subset`, the model can be fitted only to a part of the data, for more information about this functionality, see [subset](#).

Note that, unlike in `glm`, weights are **not** treated as prior weights, but as frequency weights!

Optimization and Verbosity: Using the `control` argument, the settings passed to the optimizers can be altered. This argument takes a named list. To supply user-defined starting values, use `control = list(sv=c(...))` and supply a vector containing initial values for all parameters. Optimizer-specific options include the number of iterations (`iterlim = 1000`) and convergence criteria for the BFGS- and NR-optimization (`(tol1 = 1e-5)` and `(tol2 = 1e-10)`).

Verbosity takes integer values from 0 to 4. 0, no information is printed (default). 1 prints information about 3 stages (preparation, starting values, estimation). 2 prints little information about optimization (verbosity values greater than one are passed to `print.default = verbosity - 1` of `maxBFGS` and `maxNR`). 3 prints more information about optimization. 4 prints all information about optimization.

Value

<code>call</code>	[language] function call
<code>parametrization</code>	[character] used parametrization
<code>varnames</code>	[character] components' names
<code>n.vars</code>	[numeric] vector with the number of parameters per set of predictors
<code>dims</code>	[numeric] number of components
<code>Y</code>	[numeric] used components
<code>X</code>	[numeric list] sets of predictors
<code>Z</code>	[numeric list] sets of predictors (only for the alternative parametrization)
<code>sub.comp</code>	[numeric] vector of single components
<code>base</code>	[numeric] base (only for the alternative parametrization)
<code>weights</code>	[numeric] vector of frequency weights
<code>orig.resp</code>	[DirichletRegData] the original response
<code>data</code>	[data.frame] original data
<code>d</code>	[data.frame] used data
<code>formula</code>	[Formula] expanded formula
<code>mf_formula</code>	[language] expression for generating the model frame
<code>npar</code>	[numeric] number of parameters
<code>coefficients</code>	[numeric] named vector of parameters
<code>coefnames</code>	[character] names of the parameters
<code>fitted.values</code>	[list of matrices] list containing alpha's, mu's, phi's for the observations
<code>logLik</code>	[numeric] the log-likelihood
<code>vcov</code>	[matrix] covariance-matrix of parameter estimates
<code>hessian</code>	[matrix] (observed) Hessian
<code>se</code>	[numeric] vector of standard errors
<code>optimization</code>	[list] contains details about the optimization process provided by <code>maxBFGS</code> and <code>maxNR</code>

Author(s)

Marco J. Maier

Examples

```
ALake <- ArcticLake
ALake$Y <- DR_data(ALake[,1:3])

# fit a quadratic Dirichlet regression models ("common")
res1 <- DirichReg(Y ~ depth + I(depth^2), ALake)

# fit a Dirichlet regression with quadratic predictor for the mean and
# a linear predictor for precision ("alternative")
res2 <- DirichReg(Y ~ depth + I(depth^2) | depth, ALake, model="alternative")

# test both models
anova(res1, res2)

res1
summary(res2)
```

GlacialTills

Glacial Tills

Description

Data from Aitchison (2003)

Usage

GlacialTills

Format

A data frame with 92 observations on the following 5 variables.

Red.Sandstone a numeric vector
Gray.Sandstone a numeric vector
Crystalline a numeric vector
Miscellaneous a numeric vector
Pcount a numeric vector

Source

Aitchison, J. (2003). *The Statistical Analysis of Compositional Data*. The Blackburn Press, Caldwell, NJ.

`plot.DirichletRegData` *Plot Dirichlet-Distributed Data*

Description

With this function you can plot Dirichlet-distributed data in 2, 3 and 4 dimensions.

Usage

```
## S3 method for class 'DirichletRegData'  
plot(x, dims, ticks = TRUE, ref.lines = NULL, dim.labels, a2d = list(colored =  
  TRUE, c.grid = TRUE, col.scheme = c("dims", "entropy"), entropy.contours =  
  FALSE, entropy.colors = FALSE), a3d = list(rgl = TRUE, ...), rug = TRUE,  
  reset_par = TRUE, ...)
```

Arguments

<code>x</code>	data prepared with DR_data
<code>dims</code>	select two, three, or four Dimensions of your data <code>x</code> to plot
<code>ticks</code>	display ticks?
<code>ref.lines</code>	.
<code>dim.labels</code>	a character vector giving labels for the dimensions/variables
<code>a2d</code>	a named list of settings for ternary plots (3 variables), see Details
<code>a3d</code>	a named list of settings for quaternary plots (4 variables), see Details
<code>rug</code>	display a rug for a one-dimensional plot (2 variables)
<code>reset_par</code>	reset graphical parameters of DR_data after creating a two-dimensional plot (2 variables), see Details
<code>...</code>	further graphical arguments as <code>col</code> , <code>pch</code> , <code>cex</code> , ...

Author(s)

Marco J. Maier

Examples

```
# plot of "Sand" in the Arctic Lake data set  
plot(DR_data(ReadingSkills[, 1]), main="Reading Accuracy")  
  
# ternary plot of Arctic Lake data  
plot(DR_data(ArcticLake[, 1:3]), a2d = list(colored = FALSE))
```

Reading Accuracy Data *Pammer and Kevan's Data on Reading Skills*

Description

These data provide transformed reading accuracy scores predicted by IQ and diagnosed dyslexia.

Usage

ReadingSkills
ReadDisc

Format

A data frame containing 44 observations on 3 variables.

accuracy reading accuracy score transformed to fit into (0, 1)

dyslexia a factor with the diagnosis of dyslexia (“yes” or “no”)

iq non-verbal IQ (z -scores; $\mu = 0$, $\sigma^2 = 1$)

Source

Example 3 from <http://psychology3.anu.edu.au/people/smithson/details/betareg/betareg.html>

Rocks

Aitchison's Rock Data

Description

A compilation of four datasets listed in Aitchison (2003)

Each type of rock has 25 observations – to use only a certain type of rock, see “Details”.

Usage

Rocks

Format

A data frame with 100 observations on the following 8 variables.

Albite, Blandite, Cornite, Daubite, Endite numeric vectors

depth a numeric vector

porosity a numeric vector

type a factor with levels Boxite Coxite Hongite Kongite

Source

Aitchison, J. (2003). *The Statistical Analysis of Compositional Data*. The Blackburn Press, Caldwell, NJ.

Simplex-Transformations

Transform Compositional Data for a Simplex

Description

These functions transform a matrix with three or four components to fit into a two- or three-dimensional simplex (triangle or tetrahedron).

Usage

`toSimplex(x)`

`toTernary(abc)`

`toTernaryVectors(c1, c2, c3)`

`toQuaternary(abcd)`

`toQuaternaryVectors(c1, c2, c3, c4)`

Arguments

<code>x</code>	a matrix-like object with 3 or 4 columns.
<code>abc</code>	a matrix-like object with 3 columns.
<code>abcd</code>	a matrix-like object with 4 columns.
<code>c1</code>	a numeric vector with values of the first component.
<code>c2</code>	a numeric vector with values of the second component.
<code>c3</code>	a numeric vector with values of the third component.
<code>c4</code>	a numeric vector with values of the fourth component.

Details

Most of these functions are only used internally, but sometimes it might be useful to plot “custom” ternary or quaternary graphics.

Note that, apart from `toSimplex()`, functions do not have *any* checks, so it is advisable to use this function if elements are added to plots or own graphics are created.

Value

The function returns a matrix object with coordinates in two or three dimensions

Note

In prior versions (up to 0.5-0), an unexported function `coord.trafo()` was used internally and could also be accessed via `DirichletReg:::coord.trafo()`.

If you have used this in your code, you will get a message that the function is now deprecated and will become defunct in the future. Use `toSimplex()` instead.

Author(s)

Marco J. Maier

Examples

```
# create a DirichletRegData object from the Arctic Lake data  
"to be added"
```

Index

- *Topic **datasets**
 - ArcticLake, 3
 - BloodSamples, 4
 - GlacialTills, 14
 - Reading Accuracy Data, 16
 - Rocks, 16
- *Topic **hplot**
 - plot.DirichletRegData, 15
- *Topic **manip**
 - DirichletRegData, 6
 - Simplex-Transformations, 17
- *Topic **models**
 - DirichReg, 11
- *Topic **multivariate**
 - DirichReg, 11
- *Topic **package**
 - DirichletReg-package, 2
- *Topic **regression**
 - DirichReg, 11

- AIC.DirichletRegModel
(DirichletRegModel), 9
- all.equal, 7
- anova.DirichletRegModel, 2
- ArcticLake, 3

- BIC.DirichletRegModel
(DirichletRegModel), 9
- BloodSamples, 4

- confint.DirichletRegModel
(DirichletRegModel), 9

- ddirichlet (Dirichlet), 5
- ddirichlet_R (Dirichlet), 5
- Dirichlet, 5
- DirichletReg (DirichletReg-package), 2
- DirichletReg-package, 2
- DirichletRegData, 6
- DirichletRegModel, 9

- DirichReg, 7, 11
- DR_data, 12, 15
- DR_data (DirichletRegData), 6
- drop1, 10
- drop1.DirichletRegModel
(DirichletRegModel), 9

- fitted.DirichletRegModel
(DirichletRegModel), 9

- GlacialTills, 14
- glm, 13

- lines.DirichletRegData
(plot.DirichletRegData), 15
- logLik.DirichletRegModel
(DirichletRegModel), 9

- maxBFGS, 13
- maxNR, 13

- nobs.DirichletRegModel
(DirichletRegModel), 9

- plot.DirichletRegData, 15
- predict.DirichletRegModel
(DirichletRegModel), 9
- print.DirichletRegConfint
(DirichletRegModel), 9
- print.DirichletRegData
(DirichletRegData), 6
- print.DirichletRegModel
(DirichletRegModel), 9

- rdirichlet (Dirichlet), 5
- ReadDisc (Reading Accuracy Data), 16
- Reading Accuracy Data, 16
- ReadingSkills (Reading Accuracy Data),
16
- residuals.DirichletRegModel
(DirichletRegModel), 9

Rocks, [16](#)

Simplex-Transformations, [17](#)

subset, [13](#)

summary.DirichletRegData
 (DirichletRegData), [6](#)

summary.DirichletRegModel
 (DirichletRegModel), [9](#)

toQuaternary (Simplex-Transformations),
 [17](#)

toQuaternaryVectors
 (Simplex-Transformations), [17](#)

toSimplex (Simplex-Transformations), [17](#)

toTernary (Simplex-Transformations), [17](#)

toTernaryVectors
 (Simplex-Transformations), [17](#)

update.DirichletRegModel
 (DirichletRegModel), [9](#)

update.Formula, [10](#)

vcov.DirichletRegModel
 (DirichletRegModel), [9](#)