Package ‘mixexp’

August 12, 2016

Version 1.2.5
Date 2016-8-12
Type Package
Title Design and Analysis of Mixture Experiments
Author John Lawson [aut, cre],
    Cameron Willden [aut],
    Greg Piepel [ctb]
Maintainer John Lawson <lawson@byu.edu>
Depends gdata, lattice, grid, daewr
Description Functions for creating designs for mixture experiments, making ternary contour plots, and making mixture effect plots.
License GPL (>= 2)
LazyLoad yes
LazyData yes
Repository CRAN
Repository/R-Forge/Project daewr
Repository/R-Forge/Revision 128
Repository/R-Forge/DateTimeStamp 2016-08-12 13:23:40
Date/Publication 2016-08-12 16:59:05
NeedsCompilation yes

R topics documented:

mixexp-package .................................................. 2
Burn ................................................................. 3
conmx .............................................................. 4
crvtave ........................................................... 4
cubic ............................................................... 5
DesignPoints ..................................................... 6
EffPlot ............................................................... 7
This package contains functions for creating designs for mixture experiments and making graphical display of results of mixture experiments.

Description

The mixexp package provides functions for creating mixture experiment designs in an unconstrained simplex or constrained mixture space. Functions are also provided for making ternary contour plots, pictures of constrained regions, design points, and mixture effect plots.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>mixexp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.2-1</td>
</tr>
<tr>
<td>Date:</td>
<td>2011-05-31</td>
</tr>
<tr>
<td>License:</td>
<td>GPL2.0</td>
</tr>
<tr>
<td>Dependencies:</td>
<td>gdata, lattice, grid</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
<tr>
<td>Packaged:</td>
<td>2011-05-31 19:54:07 UTC; Lawson</td>
</tr>
<tr>
<td>Built:</td>
<td>R 2.12.1; i386-pc-mingw32; 2011-05-31 19:54:08 UTC; windows</td>
</tr>
</tbody>
</table>

Index:

- conmx: example constraint matrix used as input to function crvtave
- crvtave: function for creating extreme vertices designs and centroids; this function calls Eflags, Nrows, and Vertcen
**DesignPoints**
function for plotting design points and or mixture constraint in the simplex

**Eflags**
function for calling Piepel's fortran code cnvrt to create extreme vertices designs and prints any error messages

**Effplot**
function for making mixture effect plots given a design

**MixturePlot**
function for making contour plots in simplex region given a design

**MixModel**
function for fitting mixture models to data

**ModelPlot**
function for making contour plots of an equation in an lm object created by the lm function or the MixModel function

**Nrows**
function for calling Piepel's fortran code cnvrt to create extreme vertices designs and returns the number of rows in the resulting design

**SCD**
function for creating Simplex Centroid Designs

**SLD**
function for creating Simplex Lattice Designs

**Vertcen**
function for calling Piepel's fortran code cnvrt to create extreme vertices designs and returns the resulting design

**Xvert**
function for creating extreme vertices design and centroids, this function calls crvtave

**Author(s)**
John Lawson <lawson@byu.edu> and Cameron Willden <ccwillden@gmail.com>

Maintainer: John Lawson <lawson@byu.edu>

**References**

---

**Burn**

*Data from Table 4 in Gallant,Prickett, Cesarec, and Bruck(2008)*

**Description**
This is an .rda file containing a mixture-process variable experiment with 3 mixture components and one process variable. z is the coded value of RPM, and y is average Burning rate for test pressure 500(psig)

**Usage**
data(Burn)

**Format**
An 15 x 5 data frame
Source

source

References


conmx

*Example constraint matrix from Piepel 1988*

Description

This is an .rda file containing the constraint matrix.

Usage

data(conmx)

Format

An 8 x 4 matrix

Source

source

References


crvtave

*This function creates an extreme vertices design*

Description

This function calls the function Vertcen which uses Piepel’s (1988) fortran code (cnvrt) for generating extreme vertices and centroids of linearly constrained mixture experimental regions.

Usage

crvtave( ndm, conmx)
cubic

Arguments

ndm is an integer representing the highest order of centroids requested. An overall centroid is always included, 0 indicates no other centroids will be created, 1 indicates edge centroids are requested, 2 indicates face centroids, etc.

conmx This is the matrix of constraints.

Value

vtcn This is a data frame containing the extreme vertices design. The columns are labeled x1, x2 ...xn, where n is the number of mixture variables. The last column is labeled dimen and it indicates the order of centroid where 0 is an extreme vertex, 1 is an edge centroid, 2 is a face centroid, and n is the overall centroid.

Note

This function calls the function Eflags to get error messages from cnvrt, the function Vertcen to get the extreme vertices and centroids from cnvrt, and the function Nrows to get the number of vertices and centroids from cnvrt.

Author(s)

John S. Lawson <lawson@byu.edu>

References


Examples

data(conmx)
crvtave(1,conmx)

cubic

Creates cubic terms for Scheffe' full cubic model (3)

Description

Creates cubic terms that are used by the function MixModel when fitting model (3)

Usage

cubic(a, b)
DesignPoints

Arguments

- **a**: input - vector of mixture components in a column in the data frame
- **b**: input - another vector of mixture components in a column in the data frame

Value

A vector of elementwise $a^2*b-a*b^2$ function of terms in the a and b vectors.

Author(s)

John Lawson

Description

This function plots design points and/or constraints in the simplex mixture space, given a data frame containing the design or vectors x, y, and z of the same length that contain the mixture components in the design.

Usage

```r
DesignPoints(des = NULL, nmxcmp = 3, x = NULL, y = NULL, z = NULL, x1lower = 0, x1upper = 0,
             x2lower = 0, x2upper = 0, x3lower = 0, x3upper = 0,
             cornerlabs = c("x3","x2","x1"),
             axislabs = c("x1","x2","x3"), pseudo = FALSE)
```

Arguments

- **des**: data frame containing x1, x2 and x3 coordinates of data points to be plotted
- **nmxcmp**: integer indicating the number of mixture components in the design
- **x**: vector of x3 coordinates of design points to be plotted
- **y**: vector of x2 coordinates of design points to be plotted
- **z**: vector of x1 coordinates of design points to be plotted
- **x1lower**: lower constraint on x1
- **x1upper**: upper constraint on x1
- **x2lower**: lower constraint on x2
- **x2upper**: upper constraint on x2
- **x3lower**: lower constraint on x3
- **x3upper**: upper constraint on x3
axislabs: This is a vector of text labels for the x1, x2 and x3 axis.
cornerlabs: This is a vector of text labels for the x1, x2 and x3 vertices.
pseudo: logical variable, when TRUE plot is made in pseudo component space bounded by the lower constraints of each component.

Note
This function calls MixturePlot. If either des and x,y,z are missing no design points will be plotted, and if x1lower, x1upper, etc. are all zero no constraints will be plotted. If there are more than 3 columns of mixture components in des, only the first 3 will be plotted ignoring the others.

Author(s)
John S. Lawson <lawson@byu.edu>

References

Examples
```r
dat<-SCD(3)
DesignPoints(des=dat)

x1<-c(1,0,0,5,0,.33333)
x2<-c(0,1,0,5,0,.33333)
x3<-c(0,0,1,0,5,.33333)
DesignPoints(x=x3,y=x2,z=x1)

dat<-data.frame(x1,x2,x3)
DesignPoints(des=dat)
```

EffPlot: This function creates mixture effect plots

Description
This function makes effect plots using the Cox or Piepel directions in constrained mixture space.

Usage
EffPlot(des=NULL,nfac=3,mod=1,dir=1)
Arguments
des  
  data frame containing the design points and response data for a mixture experiment. The data frame must contain the variables x1, x2 ... xn for the mixture variables, and y for the response. n must be between 2 and 12. Only effect plots for linear models can be made when the number of factors is greater than 6.
nfac  
  The number of mixture components in the model.
mod  
  an integer representing the model to be traced: 1 for a linear model, 2 for a quadratic model, and 4 for a special cubic model. For models other than these, use the ModelEff function.
dir  
  an integer representing the direction for which the effect plot is made: 1 for Piepel direction, 2 for Cox direction.

Value
PX  
  This is a matrix containing the coordinates of the effect plot traces that are plotted.

Note
This function calls the function crvave to get the design centroid from cnvrt.

Author(s)
John S. Lawson <lawson@byu.edu>

References

Examples

#Example from Li, Tolley, Lee(2010) response is perm
x1<-c(572.358.286.286.286.143.357)
x2<-c(214.428.500.357.214.500.500)
x3<-c(214.214.214.357.500.357.143)
y<-c(7.7,18.4,24.2,9.8,5.9,23.0,19.4)
des<-data.frame(x1,x2,x3,y)
EffPlot(des,2,2)

#Example from Snee, Marquart(1976)
x1<-c(.1,.1,.1,.1,.1,.1,.4,.35,.30,.1,.45,.45,.45,.45,.45,.45,.259,.259,.259,.259)
# Weed control example from Lawson & Erjavec
x1<-c(1,0,0,-5,5,0,33333,33333,33333)
x2<-c(0,1,0,-5,5,0,33333,33333,33333)
x3<-c(0,0,1,-5,5,0,33333,33333,33333)
y<-c(73,68,80,77,86,75,92,93,88)
des<-data.frame(x1,x2,x3,y)
EffPlot(des,mod=1,dir=1)

data<-as.matrix(des)
# remove the edge centroid at the top
data<-data.frame(data[2:9,])
# add two more centroids
data<-rbind(data[10,],data[10,])
# response vector
y<-c(5.75,3.69,5.33,5.68,3.85,3.83,5.88,5.87,5.23,6.54,6.82,6.41)
data<-data.frame(data[,1:3],y)
EffPlot(des,mod=3)

data<-data.frame(data[,1:3],y)
# Cornell’s example of blending pesticides for control of mites (special cubic model)
mite<-scd(point
yavg<-c(1.8,25,4,28,6,38.5,4.9,3.1,28.7,3.4,37.4,10.7,22.0,2.6,2.4,
11.1,0.8)
mite<-cbind(mite,yavg)
mite<-mite
names(mite)<-c("x1","x2","x3","x4","y")
EffPlot(des=mite,mod=4,dir=2)
Description

This function loads and runs the compiled fortran code cnvrt and prints error messages. cnvrt is Piepel’s 1988 JQT fortran code for extreme vertices designs.

Usage

Eflags(ndm,nvrr,ncon2,rtheta2)

Arguments

ndm
This is the order of centroids desired (0=none, 1=edge centroids, 2=face centroids etc.)

nvrr
This is the number of mixture variables (maximum is 12)

ncon2
This is the number of constraints (maximum is 45)

rtheta2
This is the constraint matrix stored as a vector of columns.

Value

ifa
This is the vector of error flags. A negative value for flag 1 indicates that there are inconsistent constraints. A negative value for flag2 indicates there are two many vertices and centroids, this program only works when # vertices + # centroids <=1000. A negative value for flag 3 indicates an error encountered when calling subroutine allnr.

Note

This function is called by the function crtave.

Author(s)

John S. Lawson <lawson@byu.edu>

References

Data from Etch rate experiment in Table 12.4 of Myers and Montgomery (2002)

**Description**

This is an .rda file containing a mixture experiment with 3 mixture components. The mixture components are x1, x2, and x3. The response is erate.

**Usage**

data(etch)

**Format**

An 14 x 4 data frame

**Source**

source

**References**


---

fillv

This function Creates interior points in an existing mixture design.

**Description**

This function creates interior points in a mixture design by averaging all possible pairs of design points. It duplicates SAS macro adxfill.

**Usage**

fillv(nfac, des)

**Arguments**

- **nfac**: an integer representing the number of mixture variables in the design
- **des**: a data frame containing a mixture design created by one of the functions SLD, SCD or Xvert
Author(s)
John S. Lawson <lawson@byu.edu>

Examples

# Example 1 fills interior of Simplex Lattice Design
des<-SLD(3,3)
DesignPoints(des)
des2<-fillv(3,des)
DesignPoints(des2)

# Example 2 fills interior of Simplex Centroid Design
des<-SCD(4)
fillv(4,des)

# Example 3 fills interior of Extreme vertices design
ev<-Xvert(3,uc=c(1,1,1),lc=c(0,0,0),ndm=1)
ev2<-fillv(3,ev)

fishp  Data from Cornell's famous fish patty mixture process variable experiment

Description
This is an .rda file design and response.

Usage
data(fishp)

Format
An 56 x 7 data frame

Source
source

References
**MixModel**

Fit mixture and mixture process variable models.

### Description

This function fits mixture models (1)-(4) and mixture process models (5)-(6) described in Lawson and Willden (2015) "Mixture Experiments in R, using mixexp", Journal Statistical Software http://www/jstatsoft.org/, and prints the correct R square and standard errors of model coefficients.

### Usage

```r
MixModel(frame, response, mixcomps=NULL, model, procvars=NULL)
```

### Arguments

- `frame`: a data frame containing columns with the mixture components, process variables, and responses
- `response`: a character variable containing the column name of the response variable in frame to be fit
- `mixcomps`: a character vector of column names of the mixture components in frame
- `model`: an integer in the range of 1 to 6, indicating the model to be fit:
  1. \[ y = \sum_{i=1}^{q} \beta_i x_i + \epsilon. \]
  2. \[ y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \epsilon. \]
  3. \[ y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \sum_{i=1}^{q-2} \sum_{j=i+1}^{q} \sum_{k=j+1}^{q} \delta_{ijk} x_i x_j x_k + \epsilon. \]
  4. \[ y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \sum_{i=1}^{q-2} \sum_{j=i+1}^{q} \sum_{k=j+1}^{q} \beta_{ijk} x_i x_j x_k + \epsilon. \]
  5. \[ y = \left( \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j \right) \left( \alpha_0 + \sum_{l=1}^{p} \alpha_l z_l + \sum_{l=1}^{p-1} \sum_{m=l+1}^{p} \alpha_{lm} z_l z_m \right) + \epsilon. \]
  6. \[ y = \sum_{i=1}^{q} \beta_i^{(0)} x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij}^{(0)} x_i x_j + \sum_{k=1}^{m} \left[ \sum_{i=1}^{q} \beta_i^{(1)} x_i \right] z_k + \sum_{k=1}^{m} \sum_{l=k+1}^{m} \alpha_{kl} z_k z_l + \sum_{k=1}^{m} \alpha_{kk} z_k^2 + \epsilon. \]
- `procvars`: a character vector of column names of the process variables in frame to be included in the model. Leave this out if there are no process variables in the frame

### Author(s)

John S. Lawson <lawson@byu.edu>

### References

Examples

# example from Lawson(2014), quadratic model
library(daewr)
data(pest)
mixvars<-c("x1", "x2", "x3")
MixModel(pest, "y", mixvars, 2)

# example from Myers and Montgomery(2002), special cubic model
library(mixexp)
etch<-SCD(3)
etch<-rbind(etch[1:7,], etch[1:3,], etch[7,], etch$etchx1==2/3, etch$etchx2==2/3, etch$etchx3==2/3)
erate<-c(540,330,295,610,425,330,800,560,350,260,850,710,640,460)
etch<-cbind(etch, etrate)
mixvars<-c("x1", "x2", "x3")
response<-c("erate")
MixModel(etch, response, mixvars, 4)

# example Mixture process variable model from Sahni, Pieple and Naes(2009)
library(daewr)
mixvars<-c("x1", "x2", "x3")
procvars<-c("z1", "z2")
data(MPV)
MixModel(MPV, "y", mixvars, 5, procvars)

### Kowalski Cornell and Vining Simplified model on data from Gallant et. al. (2008)
data(Burn)
testBNM<-MixModel(Burn, "y", mixcomps=c("Course", "Fine", "Binder"), model=6, procvars=c("z"))

---

**MixturePlot**

This function makes contour plots in the simplex mixture space.

### Description

This function makes contour plots in the simplex mixture space, it also can draw constraint lines and show design points.

### Usage

```
MixturePlot(x=NULL, y=NULL, z=NULL, w=NULL, des=NULL, res=400, lims=c(rep(0,6)), color.palette = heat.colors, constrts=FALSE, contrs=TRUE, n.breaks=10, levels=NULL, cols=FALSE, despts=TRUE, mod=NA, x3lab="Fraction X3", x2lab="Fraction X2", x1lab="Fraction X1", corner.labs = NULL,
```

MixturePlot

colorkey=list(dx=0.04,x0=0.95,y0=0.45,y1=0.99,add=TRUE,mode="all"), pseudo=FALSE)

Arguments

x  x3 locations for known points
y  x2 locations for known points
z  x1 locations for known points
w  y locations for known points
des  data frame with x1,x2,x3, and y locations for known points
res  number of color blocks between 0 and 1 of x
lims  vector of lower and upper constraints for x1,x2,x3
colorpalette  is the color palette to use
constrs  if TRUE constraints found in lines will be added to the graph
contrs  if TRUE contour lines will be added to the graph
n.b breaks  number of breaks between levels, this is used if levels is not specified
levels  vector of contour levels to be plotted
cols  if TRUE regions between contour lines will be colored
despts  if TRUE plots the design points in data frame des
mod  is an indicator for the model 1=linear, 2=quadratic, 4=special cubic. for other Models use the ModelEff function.
x3lab  label for the x3 axis
x2lab  label for the x2 axis
x1lab  label for the x1 axis
cornerlabs  labels for x3, x2 and x1 vertices
colorkey  a list with the location of the color key
pseudo  if pseudo=TRUE uses pseudo components to zoom in on constrained region. By default pseudo=FALSE

Author(s)

John S. Lawson <lawson@byu.edu>

References

2. See R Ternary Level Plot Function  http://www.siftp.net/index.shtml
Examples

```r
# Usage and Examples - Example from page 458 DAE with SAS
dat = data.frame(
  "x1"=c(1, 8.6, 5.3333, 3.3, 1, 1, 0, 0, 0),
  "x2"=c(0, 1.2, 5.3333, 2.5, 1.8, 0, 5, 1),
  "x3"=c(0, 1.2, 5, 33333, 5.2, 8.1, 1, 0, 5, 0),
  "y"=c(48.7, 49.5, 50.2, 52.8, 49.3, 51, 1, 52.7, 50.3, 60.7, 74.9, 64.9, 53.5, 50.6)
)
MixturePlot(dat$x3, dat$x2, dat$x1, dat$y, x3lab="Fraction x3",
  x2lab="Fraction x2", x1lab="Fraction x1", corner.labs=c("x3", "x2", "x1"),
  constrs=FALSE, contrs=TRUE, cols=TRUE, mod=2, n.breaks=9)

# Weed control example from Lawson & Erjavec
x1<-c(1, 0, 0, 5, 0, 0, 33333, 33333)
x2<-c(0, 1, 0, 5, 0, 5, 33333, 33333)
x3<-c(0, 0, 1, 0, 5, 5, 33333, 33333)
y<-c(73, 6, 80, 77, 86, 75, 92, 93, 88)
des<-data.frame(x1, x2, x3, y)
MixturePlot(des, x3lab="Fraction C", x2lab="Fraction B",
  x1lab="Fraction A", corner.labs=c("C", "B", "A"), mod=4, n.breaks=5, cols=TRUE)
```

---

**ModelEff**

This function creates mixture effect plots

**Description**

This function makes effect plots using the Cox or Piepel directions in constrained mixture space.

**Usage**

```r
ModelEff(nfac=3, mod=1, nproc=0, dir=1, ufunc=mod, dimensions = list(NULL),
  pvslice=cbind(1, 1, 1), lc=cbind(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  uc=cbind(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1))
```

**Arguments**

- **nfac**
  - The number of mixture components in the model (a number between 2 and 12)

- **mod**
  - An integer representing the model to be traced: 1 for a linear model, 2 for a quadratic model, and 3 for a cubic model, 4 for a special cubic model, 5 for a mixture process variable model consisting of a full cross of quadratic model in up to 5 mixture components and a linear model in up to 3 process variables, 6 for Kowalski, Cornell and Vining’s (2000) more parsimonious model for mixture process variable experiments. See the documentation for the MixModel function for a description of the models.

- **nproc**
  - The number of process variables in the model (a number between 1 and 3 for models 5 and 6)
ModelEff

`dir` an integer representing the direction for which the effect plot is made: 1 for Piepel direction, 2 for Cox direction.

`ufunc` A user function, this should an `lm` object created by the MixModel function. Any `lm` object will work if the terms are in the same order as the model produced by the MixModel function.

`dimensions` A vector of names of mixture components in the `lm` object.

`pvslice` A vector giving fixed values of the process variables.

`lc` A vector giving the lower bounds of the mixture components.

`uc` A vector giving the upper bounds of the mixture components.

**Value**

`px` This is a matrix containing the coordinates of the effect plot traces that are plotted.

**Note**

This function calls the function `crvtave` to get the design centroid from `cnvrt`.

**Author(s)**

John S. Lawson <lawson@byu.edu>

**References**


**Examples**

```r
#Example p. 63-65 Cornell (control of Mites)
# Four Component Mixture
mite<-SCD(4)
yavg<-c(1.8,25.4,28.6,38.5,4.9,3.1,28.7,3.4,37.4,10.7,22.0,2.6,2.4,

11.1,0.8)
mite<-cbind(mite,yavg)
miteSC<-MixModel(mite,"yavg",mixcomps=c("x1","x2","x3","x4"),model=4)
ModelEff(nfac=4,mod=4,nproc=0,dir=2,ufunc=miteSC,lc=c(0,0,0,0),uc=c(1,1,1,1))

# Cornell's (2002) Yarn elongation
x1<-c(1,1.5,.5,5,0,0,0,0,.5,.5,.5)
x2<-c(0,0,.5,.5,1,1,.5,.5,0,0,0,0)
```
ModelPlot

This function makes contour plots of a user-supplied model in the simplex mixture space.

ModelPlot

x3<-c(0,0,0,0,0,0,0,0,5,5,5,5,5,1,1,1,5,5,5,5)
y<-c(11.12.4.15,14.8,16.1,8.8,10,18,9.7,11.8,16.8,16,17.7,16.4,16.6)
elong<-data.frame(x1,x2,x3,y)
testQ2<-MixModel(elong,"y",mixcomps=c("x1","x2","x3"),model=2)
ModelEff(nfac=3,mod=2,nproc=0,dir=2,ufunc=testQ2,lc=c(0,0,0),uc=c(1,1,1))

#### Kowalski Cornell and Vining Simplified model on data from Gallant et. al. (2008)
data(burn)
testBNM<-MixModel(burn,"y",mixcomps=c("Course","Fine","Binder"),model=6,procvars=c("z"))
ModelEff(nfac=3,mod=6,nproc=1,dir=1,ufunc=testBNM,dimensions = list(NULL), pvslice=c(1),
lc=c(.403,.166,.130),uc=c(.704,.412,.210))
ModelEff(nfac=3,mod=6,nproc=1,dir=1,ufunc=testBNM,dimensions = list(NULL), pvslice=c(-1),
lc=c(.403,.166,.130),uc=c(.704,.412,.210))

---

Description

This function makes contour plots in the simplex mixture space. It also can draw constraint lines and zoom in on pseudo component region.

Usage

```r
ModelPlot(model=NULL,user.func = NULL, dimensions = list(x1=NULL,x2=NULL,x3=NULL),
slice=NULL,lims=rep(0,6), constraints = FALSE,
constraintpars = list(lty=2,lwd=2),
contour = FALSE, contourpars = list(lwd=0.5,cex.lab=1.3),
cuts = 10,at = NULL, res=300, pseudo=FALSE,
fill=FALSE, color.palette = heat.colors,
main=NULL, axislabs=c("Fraction X1","Fraction X2","Fraction X3"),
axislabpars = list(),
axisslab.offset=0,
cornerlabs = c("X1", "X2", "X3"),
cornerlabpars = list(),
grid=TRUE, gridpars = list(col='darkgrey',lty=3,lwd=0.5),
colorkey = FALSE,
labels=TRUE, label.style="align", ...)
```

Arguments

- `model` an `lm` object, `MixModel` object, or any other model object that is compatible with the predict function, which is the mixture model to be plotted.
user.func function supplied by the user that takes as arguments a dataframe called 'grid' and returns the predictions. This argument has been deprecated in favor of the model argument. Typically, this will be a wrapper function for predict() (e.g. predict(model,newdata=grid)). Additional arguments for user.func can be passed using the ellipsis argument for ModelPlot. Overrides model argument if both are specified.

dimensions list argument that specifies the mixture variables to be plotted on the ternary plot. Values must correspond to variable names from the user-supplied model.

dimension.ps list argument that specifies the mixture variables to be plotted on the ternary plot. Values must correspond to variable names from the user-supplied model.

dimensions list argument that specifies the mixture variables to be plotted on the ternary plot. Values must correspond to variable names from the user-supplied model.

slice list argument that specifies the value of fixed mixture components.

lims vector of lower and upper constraints for ternary plot components (TopLower, TopUpper, LeftLower, LeftUpper, RightLower, RightUpper).

constraints if TRUE constraints found in lims will be added to the graph.

constraint.pars list of graphical parameters controlling the appearance of the constraint lines.

contour if TRUE contour lines will be added to the graph.

contour.pars list of graphical parameters controlling the appearance of the contour lines.

cuts number of breaks between levels (used for contours if 'at' not specified).

at list of contour levels (e.g. at=c(1,3,5,10) will draw contours at those heights). Overrides cuts argument.

res resolution of the grid. Corresponds to number equally spaced values along the baseline of the simplex.

pseudo if TRUE uses pseudo components to zoom in on constrained region. Will create the smallest equilateral triangle that still contains the whole constrained region.

fill if TRUE regions between contour lines will be colored.

color.palette is the color palette to use.

main character value for main title or list containing character value and graphical parameters (e.g. main=list("main title",cex=2)).

axislabs character vector of axis labels for ternary components.

axislab.pars list of graphical parameters controlling the appearance of the axislabels.

axislab.offset numeric value that creates or eliminates space between the angled axislabels and the tickmarks. Prevents axis labels from overlapping with tickmarks. Typically, absolute value would not exceed 0.05.

cornerlabs character vector of corner labels for x1, x2 and x3 vertices.

cornerlab.pars list of graphical parameters controlling the appearance of the axis labels.

grid logical argument. If true, adds gridlines to the ternary plot.

grid.pars list of graphical parameters controlling the appearance of the gridlines.

colorkey logical or list of parameters. See levelplot documentation for more details.

labels logical argument. If true, labels contour lines.

label.style controls placement of contour labels. Choose from "mixed", "flat", or "align." See panel.levelplot documentation for more details.

... additional arguments passed to user.func
Author(s)

Cameron Willden <ccwillden@gmail.com>

References

2. See R Ternary Level Plot Function http://www.siftp.net/index.shtml

Examples

```r
# Cornell's (2002) Yarn elongation
x1<-c(1,1,.5,.5,5,0,0,0,0,0,0,.5,.5)
x2<-c(0,.5,.5,5,1,.5,.5,0,0,0,0,0)
x3<-c(0,0,0,0,0,0,5,.5,1,1,.5,.5)
y<-c(11.2,4.15,14.8,16.1,18.8,10,18.9,7,11.8,16.8,16,17.7,16.4,16.6)
elong<-data.frame(x1,x2,x3,y)
testQ<-lm(y~x1+x2+x3+x1:x2+x1:x3+x2:x3, data=elong)
ModelPlot(model = testQ, dimensions = list(x=1"x1", x2="x2", x3="x3"),
main="Thread Elongation", constraints=FALSE, contour=TRUE,
at=c(12, 13, 14, 15, 16, 17), fill=FALSE,
axislabs=c("X1", "X2", "X3"),
cornerlabs = c("X1", "X2", "X3"), pseudo=FALSE)

# Cornell's famous fish Patty experiment
data(fishp)
fishmodR<-MixModel(fishp, "y", mixcomps=c("x1","x2","x3"), model=5, procvars=c("z1","z2","z3"))
ModelPlot(fishmodR, dimensions = list(x=1"x1", x2="x2", x3="x3"),
slice = list(process.vars=c(z1=-1, z2=-1, z3=-1)), main="z1=-1, z2=-1, z3=-1",
constraints=FALSE, contour=TRUE, cuts=10, fill=FALSE,
axislabs=c("Fraction X1", "Fraction X2", "Fraction X3"),
cornerlabs = c("X1", "X2", "X3"), pseudo=FALSE)

#### Kowalski Cornell and Vining Simplified model on data from Gallant et. al. (2008)
data(Burn)
testBNM<-MixModel(Burn,"y",mixcomps=c("Course","Fine","Binder"),model=6,procvars=c("z"))
ModelPlot(testBNM, dimensions = list(x=1"Course", x2="Fine", x3="Binder"),
slice = list(process.vars=c(z=1)), lims=c(.403,.704,.166,.467,.130,.431), main="z=1",
constraints=TRUE, contour=TRUE, cuts=5, fill=FALSE,
axislabs=c("Fraction Course", "Fraction Fine", "Fraction Binder"),
cornerlabs = c("Course", "Fine", "Binder"), pseudo=TRUE)

ModelPlot(testBNM, dimensions = list(x=1"Course", x2="Fine", x3="Binder"),
slice = list(process.vars=c(z=-1)), lims=c(.403,.704,.166,.467,.130,.431), main="z=-1",
constraints=TRUE, contour=TRUE, cuts=5, fill=FALSE,
axislabs=c("Fraction Course", "Fraction Fine", "Fraction Binder"),
cornerlabs = c("Course", "Fine", "Binder"), pseudo=TRUE)
```
**Nrows**  

*Loads compiled fortran in shared file cnvrt and returns the number of rows in the resulting design*

---

**Description**

This function loads and runs the compiled fortran code cnvrt. cnvrt is Piepel’s 1988 JQT fortran code for extreme vertices designs.

**Usage**

```
Nrows(ndm,nvrr,ncon2,rtheta2)
```

**Arguments**

- **ndm**: This is the order of centroids desired (0=none, 1=edge centroids, 2=face centroids etc.)
- **nvrr**: This is the number of mixture variables (maximum is 12)
- **ncon2**: This is the number of constraints (maximum is 45)
- **rtheta2**: This is the constraint matrix stored as a vector of columns.

**Value**

```
nvrtr
```

```
nvrtr
```

This is the number of rows in rxvt the matrix of extreme vertices and centroids.

**Note**

This function is called by the function crtave.

**Author(s)**

John S. Lawson <lawson@byu.edu>

**References**

This function creates simplex centroid mixture designs

Description

This function creates simplex centroid designs in unconstrained mixture experiment space.

Usage

SCD(fac)

Arguments

fac This is the number of factors

Value

SC This is a data frame containing the simplex centroid design. The columns are labeled x1, x2 ...xn, where n is the number of mixture variables.

Author(s)

John S. Lawson <lawson@byu.edu>

References


Examples

SCD(3)

des<-SCD(5)

des<-SCD(12)
This function creates simplex lattice mixture designs

Description

This function creates simplex lattice designs in unconstrained mixture experiment space.

Usage

SLD(fac, lev)

Arguments

fac
This is the number of factors, this must be between 2 and 12
lev
This is the number of levels, this must be between 2, and 5.

Value

SL
This is a data frame containing the simplex lattice design. The columns are labeled x1, x2 ...xn, where n is the number of mixture variables.

Author(s)

John S. Lawson <lawson@byu.edu>

References


Examples

des<-SLD(3, 2)
des<-SLD(4, 3)
SneeMq

Data from Snee and Marquart's Screening Experiment with constrained mixture components

Description

This is an .rda file design and response.

Usage

data(SneeMq)

Format

An 16 x 9 data frame

Source

source

References


Vertcen

Loads compiled fortran in shared file cnvrt

Description

This function loads and runs the compiled fortran code cnvrt. cnvrt is Piepel’s 1988 JQT fortran code for extreme vertices designs.

Usage

Vertcen(ndm,nvrr,ncon2,rtheta2)

Arguments

ndm
This is the order of centroids desired (0=none, 1=edge centroids, 2=face centroids etc.)

nvrr
This is the number of mixture variables (maximum is 12)

ncon2
This is the number of constraints (maximum is 45)

rtheta2
This is the constraint matrix stored as a vector of columns.
**Value**

\texttt{rxvt} 

This is the matrix of vertices and centroids stored as a vector of columns.

**Note**

This function is called by the function 	exttt{crtave}.

**Author(s)**

John S. Lawson <lawson@byu.edu>

**References**


---

**Description**

This function calls the function 	exttt{crvtave} to create an extreme vertices design in a constrained mixture space. If there are only three factors the function 	exttt{DesignPoints} is called to plot the results.

**Usage**

\[
\texttt{Xvert(nfac=3, uc=c(0,0), lc=c(0,0), nlc=0, lb=c(0,0), ub=c(0,0), coef, ndm=0, plot=TRUE, cornerlabs = c("x1","x2","x3"), axislabs = c("x1","x2","x3"), pseudo=TRUE)}
\]

**Arguments**

- \texttt{nfac} 
  an integer representing the number of mixture variables in the design. Maximum \texttt{nfac}=12
- \texttt{uc} 
  a vector of length \texttt{nfac} containing upper constraints on each mixture component
- \texttt{lc} 
  a vector of length \texttt{nfac} containing lower constraints on each mixture component
- \texttt{nlc} 
  the number of linear constraints, the default is zero
- \texttt{lb} 
  a vector of length \texttt{nlc} containing the lower bounds for the linear constraints
- \texttt{ub} 
  a vector of length \texttt{nlc} containing the upper bounds for the linear constraints
- \texttt{coef} 
  an \texttt{nlc} by \texttt{nfac} matrix containing the coefficients of the components of the linear constraints
- \texttt{ndm} 
  an integer representing the highest order of centroids requested. An overall centroid is always included, 0 indicates no other centroids will be created, 1 indicates edge centroids are requested, etc.
plot  a logical variable indicating whether a plot of the design is desired when there are only 3 components. Default is TRUE

cornerlabs  This is a vector of text labels for the x1, x2 and x3 vertices. Use when there are only 3 components for plotting.

axislabs  This is a vector of text labels for the x1, x2 and x3 axis. Use when there are only 3 components for plotting.

pseudo  logical variable, when TRUE plot in pseudo component space when there are lower constraints.

Note
This function calls crvtave. If the number of factors is 3, the function DesignPoints is called to graph the results.

Author(s)
John S. Lawson <lawson@byu.edu>

References


Examples

```r
# Polvoron Example from Lawson
des<-Xvert(3,uc=c(.8,.95,.50),lc=c(0,.1,.05),ndm=1,plot=FALSE)

#Snee Marquardt(1976) example
Xvert(8,uc=c(.45,.50,.10,.4,.6,.2,.05,.05),lc=c(.1,.05,0,.1,.05,0,0),ndm=0)

# Example page 465
exvert<-Xvert(4,uc=c(.188,.128,.438,.438),lc=c(.124,.064,.374,.374),ndm=2)

# Example from Piepel 1988
coef<-matrix(c(.85,.91,.7,0,1),ncol=3,byrow=TRUE)
des<-Xvert(3,lc=c(.1,.1,0),uc=c(.5,.7,.7),nlc=2,lb=c(.9,.4),ub=c(.95,0),coef,ndm=1,plot=FALSE)
```
Index

*Topic **datagen**
  crvtave, 4
  Fillv, 11
  SCD, 22
  SLD, 23
  Xvert, 25

*Topic **datasets**
  Burn, 3
  conmx, 4
  etch, 11
  fishp, 12
  Sneemq, 24

*Topic **hplot**
  DesignPoints, 6
  EffPlot, 7
  MixturePlot, 14
  ModelEff, 16
  ModelPlot, 18

*Topic **interface**
  Eflags, 9
  Nrows, 21
  Vertcen, 24

*Topic **package**
  mixexp-package, 2

*Topic **regression**
  MixModel, 13

Burn, 3
conmx, 4
crvtave, 4
cubic, 5

DesignPoints, 6

EffPlot, 7
Eflags, 9
etch, 11

Fillv, 11
fishp, 12