Package ‘cusp’

August 17, 2024

Type Package

Title Cusp-Catastrophe Model Fitting Using Maximum Likelihood

Version 2.3.8

Imports stats, graphics, grDevices, utils

LazyData yes

NeedsCompilation yes

Suggests plot3D

Description Cobb's maximum likelihood method for cusp-catastrophe modeling (Grasman, van der Maas, and Wagenmakers (2009) <doi:10.18637/jss.v032.i08>; Cobb (1981), Behavioral Science, 26(1), 75-78). Includes a cusp() function for model fitting, and several utility functions for plotting, and for comparing the model to linear regression and logistic curve models.

License GPL-2

Repository CRAN

Author Raoul P. P. P. Grasman [aut, cre, cph] (<https://orcid.org/0000-0001-7458-1272>)

Maintainer Raoul P. P. P. Grasman <rgrasman@uva.nl>

Date/Publication 2024-08-17 07:10:02 UTC

Contents

`cusp-package` *Cusp Catastrophe Modeling*

Description

Fits cusp catastrophe to data using Cobb's maximum likelihood method with a different algorithm. The package contains utility functions for plotting, and for comparing the model to linear regression and logistic curve models. The package allows for multivariate response subspace modeling in the sense of the GEMCAT software of Oliva et al.

Details

Package:	cusp
Type:	Package
Version:	2.0
Date:	2008-02-14
License:	GNU GPL v2 (or higher)

This package helps fitting Cusp catastrophe models to data, as advanced in Cobb et al. (1985). The main functions are

`cusp`	Fit Cobb's Cusp catastrophe model; see example below.
`summary.cusp`	Summary statistics of cusp model fit.
`confint.cusp`	Confidence intervals for parameter estimates
`plot.cusp`	Diagnostic plots for cusp model fit
`cusp3d`	3D graphical display of cusp model fit (experimental).
`dcusp`	Density of Cobb's cusp distribution
`pcusp`	Cumulative probability function of Cobb's cusp distribution
`qcusp`	Quantile function of Cobb's cusp distribution
`rcusp`	Sample from Cobb's cusp distribution.
`cusp.logist`	Fit logistic model for bifurcation testing (experimental)

Author(s)

Raoul Grasman <rgrasman@uva.nl>

References

L. Cobb and S. Zacks (1985) *Applications of Catastrophe Theory for Statistical Modeling in the Biosciences (article)*, Journal of the American Statistical Association, 392:793–802.

P. Hartelman (1996). *Stochastic Catastrophe Theory.* Unpublished PhD-thesis.

H. L. J. van der Maas, R. Kolstein, and J van der Pligt (2003). *Sudden Transitions in Attitudes*, Sociological Methods and Research, 32:125-152.

Oliva, DeSarbo, Day, and Jedidi. (1987) *GEMCAT : A General Multivariate Methodology for Estimating Catastrophe Models*, Behavioral Science, 32:121-137.

R. P. P. P. Grasman, H. L. J. van der Maas, and E-J. Wagenmakers (2009). *Fitting the Cusp Catastrophe in R: A cusp Package Primer.* Journal of Statistical Software 32(8), 1-28. URL https://www.jstatsoft.org/v32/i08/.

Examples

```
set.seed(123)
# fitting cusp to cusp data
x <- rcusp(100, alpha=0, beta=1)
fit <- cusp(y ~ x, alpha ~ 1, beta ~ 1)
print(fit)

# example with regressors
## Not run:
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)
print(fit)
summary(fit)
plot(fit)
cusp3d(fit)

## End(Not run)

# use of OK
npar <- length(fit$par)
## Not run:
while(!fit$OK) # refit if necessary until convergence is OK
    fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data, start=rnorm(npar))

## End(Not run)

## Not run:
# example 1 from paper
data(attitudes)
data(attitudeStartingValues)
```

```
fit.attitudes <- cusp(y ~ Attitude, alpha ~ Orient + Involv, beta ~ Involv,
 data = attitudes, start=attitudeStartingValues)

summary(fit.attitudes)
plot(fit.attitudes)
cusp3d(fit.attitudes, B = 0.75, Y = 1.35, theta = 170, phi = 30, Yfloor = -9)

## End(Not run)
```

`attitudes` *Multistability in political attitudes*

Description

Data set reflecting bistability in political attitudes

Usage

```
data(attitudes)
data(attitudeStartingValues)
```

Format

A data frame with 1387 observations on the following 3 variables.

`Orient` a numeric vector

`Involv` a numeric vector

`Attitude` a numeric vector

The format of attitudeStartingValues is: num [1:7] 0.153 -0.453 -0.097 -0.124 -0.227 ...

Details

The data set was taken from (van der Maas, Kolstein, & van der Pligt, 2003). It concerns attitudinal response transitions with respect to the statement “The government must force companies to let their workers benefit from the profit as much as the shareholders do”. Responses of some 1387 Dutch respondents are included who indicated their level of agreement with this statement on a 5 point scale (1 = total ly agree, 5 = total ly disagree). As a normal factor political orientation (measures on a 10 point scale from 1 = left wing to 10 = right wing) was used. As a bifurcation factor the total score on a 12 item political involvement scale was used. The theoretical social psychological details are discussed in (van der Maas et al. 2003).

The starting values provided here for a cusp analysis of the `attitude` data set give proper convergence in one run. They were found after many trial starting values that yielded improper convergence.

Source

van der Maas HLJ, Kolstein R, van der Pligt J (2003). Sudden Transitions in Attitudes. Sociological Methods & Research, 23(2), 125152.

References

van der Maas HLJ, Kolstein R, van der Pligt J (2003). Sudden Transitions in Attitudes. Sociological Methods & Research, 23(2), 125152.

Examples

```
data(attitudes)
data(attitudeStartingValues)
## Not run:
fit <- cusp(y ~ Attitude,
alpha ~ Orient + Involv,
beta ~ Involv,
data = attitudes, start=attitudeStartingValues)

## End(Not run)
## maybe str(attitudeStartingValues) ; plot(attitudeStartingValues) ...
```

`cusp` *Fit a Cusp Catastrophe Model to Data*

Description

This function fits a cusp catastrophe model to data using the maximum likelihood method of Cobb. Both the state variable may be modeled by a linear combination of variables and design factors, as well as the normal/asymmetry factor `alpha` and bifurction/splitting factor `beta`.

Usage

```
cusp(formula, alpha, beta, data, weights, offset, ..., control =
    glm.control(), method = "cusp.fit", optim.method = "L-BFGS-B", model = TRUE,
    contrasts = NULL)
```

Arguments

`formula`	`formula` that models the canonical state variable
`alpha`	`formula` that models the canonical normal/asymmetry factor
`beta`	`formula` that models the canonical bifurcation/splitting factor
`data`	`data.frame` that contains all the variables named in the formulas
`weights`	vector of weights by which each data point is weighted (experimental)
`offset`	vector of offsets for the data (experimental)
`...`	named arguments that are passed to `optim`
`control`	`glm.control` object, currently unused
`method`	string, currently unused
`optim.method`	string passed to `optim` to choose the optimization algorithm
`model`	should the model matrix be returned?
`contrasts`	matrix of `contrasts`, experimental

Details

`cusp` fits a cusp catastrophe model to data. Cobb's definition for the canonical form of the stochastic cusp catastrophe is the stochastic differential equation

$$dY_t = (\alpha + \beta Y_t - Y_t^3)dt + dW_t.$$

The stationary distribution of the 'behavioral', or 'state' variable Y, given the control parameters α ('asymmetry' or 'normal' factor) and β ('bifurcation' or 'splitting' factor) is

$$f(y) = \Psi \exp(\alpha y + \beta y^2/2 - y^4/4),$$

where Ψ is a normalizing constant.

The behavioral variable and the asymmetry and bifurcation factors are usually not directly related to the dependent and independent variables in the data set. These are therefore used to predict the state variable and control parameters:

$$y_i = w_0 + w_1 Y_{i1} + \cdots + w_p Y_{ip}$$

$$\alpha_i = a_0 + a_1 X_{i1} + \cdots + a_p X_{ip}$$

$$\beta_i = b_0 + b_1 X_{i1} + \cdots + b_q X_{iq}$$

in which the a_j's, b_j's, and w_j's are estimated by means of maximum likelihood. Here, the Y_{ij}'s and X_{ij}'s are variables constructed from variables in the data set. Variables predicting the α's and β's need not be the same.

The state variable and control parameters can be modeled by specifying a model `formula`:

`y  model`,

`alpha  model`,

`beta  model`,

in which `model` can be any valid `formula` specified in terms of variables that are present in the `data.frame`.

Value

List with components

`coefficients`	Estimated coefficients
`rank`	rank of Hessian matrix
`qr`	`qr` decomposition of the Hessian matrix
`linear.predictors`	two column matrix containing the α_i's and β_i's for each case
`deviance`	sum of squared errors using Delay convention
`aic`	AIC
`null.deviance`	variance of canonical state variable
`iter`	number of optimization iterations

`weights`	weights provided through weights argument
`df.residual`	residual degrees of freedom
`df.null`	degrees of freedom of constant model for state variable
`y`	predicted values of state variable
`converged`	convergence status given by `optim`
`par`	parameter estimates for `qr` standardized data
`Hessian`	Hessian matrix of negative log likelihood function at minimum
`hessian.untransformed`	Hessian matrix of negative log likelihood for `qr` standardized data
`code`	`optim` convergence indicator
`model`	list with model design matrices
`call`	function call that created the object
`formula`	list with the formulas
`OK`	logical. `TRUE` if Hessian matrix is positive definite at the minimum obtained
`data`	original data.frame

Author(s)

Raoul Grasman

References

See cusp-package

See Also

`cusp-package`.

`summary.cusp` for summaries and model assessment.

The generic functions `coef`, `effects`, `residuals`, `fitted`, `vcov`.

`predict` for estimated values of the control parameters $\alpha[i]$ and $\beta[i]$,

Examples

```
set.seed(123)
# example with regressors
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)
print(fit)
summary(fit)
## Not run:
plot(fit)
cusp3d(fit)
```

```
## End(Not run)

# useful use of OK
## Not run:
while(!fit$OK)
    fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data,
            start=rnorm(fit$par)) # use different starting values

## End(Not run)
```

`cusp.bifset` *compute normal/symmetry factor borders of bifurcation set of cusp catastrophe*

Description

Given bifurcation/splitting factor values this function computes the border values of the normal/symmetry factor for the bifurcation set of the cusp catastrophe.

Usage

```
cusp.bifset(beta)
```

Arguments

`beta` values of the bifurcation/splitting factor at which the border values of the normal/symmetry factor is computed

Value

Matrix with columns named `beta`, `alpha.l`, `alpha.u`. The latter two columns give respectively the lower and upper border values of the normal/symmetry factor. Negative values of `beta` give `NaN` values for the normal factor.

Author(s)

Raoul Grasman

References

See cusp-package

See Also

cusp-package

Examples

```
cusp.bifset(-3:3)
```

`cusp.extrema` *Locate Extrema of Cusp Catastrophe Potential Function*

Description

This function computes the locations of the extrema of the cusp catastrophe potential function.

Usage

```
cusp.extrema(alpha, beta)
```

Arguments

`alpha`	(single) value of normal/symmetry factor
`beta`	(single) value of bifurcation/splitting factor

Details

The locations are determined by computing the solutions to the equation

$$\alpha + \beta X - X^3 = 0.$$

Value

Ordered vector with locations of extremes.

Note

Use `Vectorize` to allow for array input.

Author(s)

Raoul Grasman

References

http://www.scholarpedia.org/article/Cusp_bifurcation

See Also

`cusp.bifset`

Examples

```
# simple use
cusp.extrema(2,3)

# using vectorize to allow for array input;
# returns a matrix with locations in each column
Vectorize(cusp.extrema)(-3:3, 2)
```

cusp.logist *Fit a Logistic Surface Model to Data*

Description

This function fits a logistic curve model to data using maximum likelihood under the assumption of normal errors (i.e., nonlinear least squares). Both the response variable may be modeled by a linear combination of variables and design factors, as well as the normal/asymmetry factor `alpha` and bifurcation/splitting factor `beta`.

Usage

```
cusp.logist(formula, alpha, beta, data, ..., model = TRUE, x =
                 FALSE, y = TRUE)
```

Arguments

`formula`, `alpha`, `beta`
: `formula`s for the response variable and the regression variables (see below)

`data`
: `data.frame` containing n observations of all the variables named in the formulas

`...`
: named arguments that are passed to `nlm`

`model`, `x`, `y`
: logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, and the response are returned.

Details

A nonlinear regression is carried out of the model

$$y_i = \frac{1}{1 + \exp(-\alpha_i/\beta_i^2)} + \epsilon_i$$

for $i = 1, 2, \ldots, n$, where

$$y_i = w_0 + w_1 Y_{i1} + \cdots + w_p Y_{ip}$$

$$\alpha_i = a_0 + a_1 X_{i1} + \cdots + a_p X_{ip}$$

$$\beta_i = b_0 + b_1 X_{i1} + \cdots + b_q X_{iq}$$

in which the a_j's, and b_j's, are estimated. The Y_{ij}'s are variables in the data set and specified by `formula`; the X_{ij}'s are variables in the data set and are specified in `alpha` and `beta`. Variables in `alpha` and `beta` need not be the same. The w_j's are estimated implicitly using concentrated likelihood methods, and are not returned explicitly.

Value

List with components

- `minimum` Objective function value at minimum
- `estimate` Coordinates of objective function minimum
- `gradient` Gradient of objective function at minimum.
- `code` Convergence code returned by `optim`
- `iterations` Number of iterations used by `optim`
- `coefficients` A named vector of estimates of a_j, b_j's
- `linear.predictors` Estimates of α_i's and β_i's.
- `fitted.values` Predicted values of y_i's as determined from the `linear.predictors`
- `residuals` Residuals
- `rank` Numerical rank of matrix of predictors for α_i's plus rank of matrix of predictors for β_i's plus rank of matrix of predictors for the y_i's.
- `deviance` Residual sum of squares.
- `logLik` Log of the likelihood at the minimum.
- `aic` Akaike's information criterion
- `rsq` R Squared (proportion of explained variance)
- `df.residual` Degrees of freedom for the residual
- `df.null` Degrees of freedom for the Null residual
- `rss` Residual sum of squares
- `hessian` Hessian matrix of objective function at the minimum if `hessian=TRUE`.
- `Hessian` Hessian matrix of log-likelihood function at the minimum (currently unavailable)
- `qr` QR decomposition of the `hessian` matrix
- `converged` Boolean indicating if optimization convergence is proper (based on exit code `optim`, gradient, and, if `hessian=TRUE` eigen values of the hessian).
- `weights` `weights` (currently unused)
- `call` the matched call
- `y` If requested (the default), the matrix of response variables used.
- `x` If requested, the model matrix used.
- `null.deviance` The sum of squared deviations from the mean of the estimated y_i's.

Author(s)

Raoul Grasman

References

Hartelman PAI (1997). *Stochastic Catastrophe Theory.* Amsterdam: University of Amsterdam, PhD thesis.

See Also

summary.cusp

---

cusp.nc *Calculate the Normalizing Constant of Cobb's Cusp Density*

Description

A family of functions that return the normalization constant for the cusp density given the values of the bifurcation and asymmetry parameters (default), or returns the moment of a specified order (cusp.nc).

Usage

```
cusp.nc(alpha, beta, mom.order = 0, ...)
cusp.nc.c(alpha, beta, ..., keep.order = TRUE)
cusp.nc.C(alpha, beta, subdivisions = 100, rel.tol = .Machine$double.eps^0.25,
    abs.tol = rel.tol, stop.on.error = TRUE, aux = NULL, keep.order = TRUE)
cusp.nc.vec(alpha, beta, ..., keep.order = FALSE)
```

Arguments

alpha	the asymmetry parameter in Cobb's cusp density (see cusp)
beta	the bifurcation parameter in Cobb's cusp density (see cusp)
mom.order	the moment order to be computed (see details below)
subdivisions, rel.tol, abs.tol, stop.on.error, aux	Arguments used by the internal integration routine of R (see integrate).
keep.order	Logical, that indicates whether the order of the output should be the same as the order of the input.
...	Extra arguments in cusp.nc.c that are passed to cusp.nc.C.

Details

The function cusp.nc returns Ψ if mom.order = 0 and Ψ times the moment of order mom.order otherwise.

The function cusp.nc is internally used if the C-routine symbol "cuspnc" is not loaded. The functions cusp.nc.c and cusp.nc.C call this C routine, which is considerably faster than cusp.nc.

These functions are not intended to be called directly by the user.

Value

`cusp.nc`, `cusp.nc.c`, `cusp.nc.vec` return a numeric vector of the same length as `alpha` and `beta` with normalizing constants, or the indicated moments times the normalization constant (`cusp.nc` only).

`cusp.nc.C` returns a list with vectors with the results obtained from `integrate`. `cusp.nc.c` first sorts the input in such a way that the numerical integrals can be evaluated more quickly than in arbitrary order

Author(s)

Raoul Grasman

See Also

`pcusp`, `dcusp`

---

cusp.nlogLike *Negative log-likelihood for Cobb's cusp density*

Description

(Negative) log-likelihood for Cobb's cusp probability density function used by `cusp`. This function is not to be called by the user. See `help(cusp)`.

Usage

```
cusp.nlogLike(p, y, X.alpha, X.beta = X.alpha, ..., verbose = FALSE)
cusp.nlogLike.c(p, y, X.alpha, X.beta = X.alpha, ..., verbose = FALSE)
cusp.logLike(p, x, verbose = FALSE)
```

Arguments

`p`	parameter vector
`x`	vector of observed values for the state variable in the cusp (`cusp.nlogLike` only)
`y`	design matrix predicting state values at which the likelihood is evaluated
`X.alpha`	design matrix predicting alpha in the model
`X.beta`	design matrix predicting beta in the model
`...`	unused extra arguments
`verbose`	logical, if `TRUE` the value of the parameters are printed to the console

Details

`cusp.nlogLike` is the R version of the corresponding C function wrapped by `cusp.nlogLike.c`

These functions are not intended to be called directly by the user.

Value

The value of the negative log-likelihood function (`cusp.nlogLike`, `cusp.nlogLike.c`), the value of the log-likelihood function (`cusp.logLike`).

Note

The functions are not to be called by the user directly.

Author(s)

Raoul Grasman

References

See cusp-package

See Also

cusp, cusp-package

`cusp3d` *Generate 3D plot of Cusp Catastrophe Model Fit*

Description

This function generates a 3D display of the fit (object) of a cusp model.

Usage

```
cusp3d(y, alpha = if (!missing(y) && is.list(y)) y$lin[, "alpha"],
    beta = if (!missing(y) && is.list(y)) y$lin[, "beta"], w = 0.03,
    theta = 170, phi = 35, B = 4, Y = 3, Yfloor = -15,
    np = 180, n.surface = 30, surface.plot = TRUE,
    surf.alpha = 0.75, surf.gamma = 1.5, surf.chroma = 35, surf.hue = 240,
    surf.ltheta = 0, surf.lphi = 45, ...)
```

Arguments

`y`	object returned by `cusp` or a vector of observed state values
`alpha`	vector of normal/symmetry factor values corresponding to the state values in `y`
`beta`	vector of bifurcation/splitting factor values corresponding to the state values in `y`
`w`	number that specifies the size of the data points plotted on the cusp surface
`theta, phi`	Angles defining the viewing direction: `theta` gives the azimuthal direction, and `phi` the colatitude.
`B`	range of the splitting factor axis

`Y`	range of the state variable axis
`Yfloor`	location on state variable axis where the control surface is plotted
`np`	factor that determines the fineness of the drawing
`n.surface`	factor that determines the fineness of the rendered surface
`surface.plot`	plot the surface?
`surf.alpha`	transparency level of rendered surface
`surf.gamma`	factor that determines the shading of surface facets (`surf.gamma`<1 diminishes shading, `surf.gamma`>1 exaggerates shading)
`surf.chroma`, `surf.hue`	chroma and hue of surface color (see `hcl`)
`surf.ltheta`, `surf.lphi`	the surface is shaded as though it was being illuminated from the direction specified by azimuth `surf.ltheta` and colatitude `surf.lphi`
`...`	named parameters that are passed to `persp`

Details

This function is experimental.

Value

`cusp3d` returns the viewing transformation matrix, say `VT`, a 4 x 4 matrix suitable for projecting 3D coordinates (x,y,z) into the 2D plane using homogeneous 4D coordinates (x,y,z,t). It can be used to superimpose additional graphical elements on the 3D plot, by lines() or points(), using the simple function trans3d().

Note

Currently still somewhat buggy.

Author(s)

Raoul Grasman

References

See `cusp-package`

See Also

`persp`, `plot.cusp`, `cusp3d.surface`

Examples

```
set.seed(123)
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)

cusp3d(fit)
```

---

cusp3d.surface *Generate 3D plot of the Cusp surface*

Description

This function generates a 3D display of the cusp equilibrium surface.

Usage

```
cusp3d.surface(alpha = c(-5, 5), beta = c(-3, 3), y = 41,
xlim = range(alpha), ylim = range(beta), zlim = c(-5, 4),
xlab = expression(alpha), ylab = expression(beta), zlab = "equilibrium states",
main = NULL, sub = NULL, phi = 20, theta = 160,
r = sqrt(3), d = 1, scale = TRUE, expand = 1, hue = 240,
chroma = 35, surf.alpha = 0.75, gamma = 1.5, bcol = NA,
lcol = "gray", ltheta = 90, lphi = 70, box = TRUE,
axes = FALSE, nticks = 5, ticktype = "simple", floor.lines = TRUE, ...)
```

Arguments

`alpha`	numeric 2-vector specifying the normal/symmetry factor axis range
`beta`	numeric 2-vector specifying the bifurcation/splitting factor axis range
`y`	numeric specifying the iso contours used to render the surface (see details below)
`xlim, ylim, zlim`	numeric 2-vectors (see `persp`)
`xlab, ylab, zlab, main, sub`	strings (see `persp`)
`phi, theta`	numeric, determine viewing direction (see `persp`)
`r`	numeric, distance to center of the plotting box (see `persp`)
`d`	numeric, strength of perspective transformation (see `persp`)
`scale, expand`	logical, see `persp`
`hue, chroma, surf.alpha`	hue, chroma and alpha (transparency) of the surface segments (see `hcl`)
`gamma`	gamma for shading of surface (see `cusp3d`)

`bcol`	color, `NA`, or string `"surface"`. Color of the border of each surface element; `NA` gives transparent borders; `"surface"` tries to hide the border as much as possible by giving it the same color as the surface segment.
`lcol`	color of the lines on the floor of the plotting cube
`ltheta, lphi`	numeric, direction of illumination of the surface (similar to `persp`)
`box, axes, nticks, ticktype`	(see `persp`)
`floor.lines`	logical, if `TRUE` (default) iso-contours are projected on the floor of the plotting cube (revealing the bifurcation set)
`...`	extra arguments that are passed to `lines` and `polygon`

Details

If `y` has length 1, it is interpreted as the number of contours. Otherwise it is interpreted as a vector of contour levels from which the surface must be determined. If `y` is a number, the exact range of `y` is determined by the ranges of `alpha` and `beta` through the cusp equilibrium equation below.

The surface is constructed from the iso-contours of the cusp equilibrium surface that makes up the solutions to

$$\alpha + \beta * y - y^3 = 0$$

as a (multi-)function of the asymmetry variable α and bifurcation variable β. For each possible solution y the iso-contours are given by the equation

$$\alpha = (\beta * y - y^3)/y,$$

which are linear in β. For each value of y the values of $alpha$ are determined for the end points of the $beta$ range specified by `beta`. The two 3D coordinates (α, β, y) are projected onto the 2D canvas using the `persp` transformation matrix and used for drawing the lines and polygons.

Value

`cusp3d.surface` returns the viewing transformation matrix, say `VT`, a 4 x 4 matrix suitable for projecting 3D coordinates (x,y,z) into the 2D plane using homogeneous 4D coordinates (x,y,z,t). It can be used to superimpose additional graphical elements on the 3D plot, by lines() or points(), using the simple function trans3d().

Note

This function is an alternative to `cusp3d` which uses a different method of rendering and also plots fitted points on the surface.

Author(s)

Raoul Grasman

References

See `cusp-package`, `cusp3d`

See Also

persp, plot.cusp

Examples

```
## Not run:
p = cusp3d.surface(chroma=40,lcol=1,surf.alpha=.95,phi=30,theta=150,
bcol="surface",axes=TRUE,main="Cusp Equilibrium Surface")
lines(trans3d(c(5,5), c(3,3), c(-5,4), p), lty=3) # replot some of the box outlines
lines(trans3d(c(-5,5), c(3,3), c(4,4), p), lty=3)

## End(Not run)
```

---

dcusp *Cobb's Cusp Distribution*

Description

Functions for the cusp distribution.

Usage

```
dcusp(y, alpha, beta)
pcusp(y, alpha, beta, subdivisions = 100, rel.tol = .Machine$double.eps^0.25,
      abs.tol = rel.tol, stop.on.error = TRUE, aux = NULL, keep.order = TRUE)
qcusp(p, alpha, beta)
rcusp(n, alpha, beta)
```

Arguments

y	vector of quantiles
p	vector of probabilities
n	number of observations.
alpha	normal/asymmetry factor value of cusp density
beta	bifurcation/splitting factor value of cusp density
subdivisions	See cusp-package.
rel.tol	See cusp-package.
abs.tol	See cusp-package.
stop.on.error	See cusp-package.
aux	See cusp-package.
keep.order	logical. If true the order of the output values is the same as those of the input values y

Details

The cusp distribution is defined by

$$f(y) = \Psi \exp(\alpha y + \beta y^2/2 - y^4/4),$$

where Ψ is the normalizing constant.

`rcusp` uses rejection sampling to generate samples.

`qcusp` implements binary search and is rather slow.

Value

`dcusp` gives the density function, `pcusp` gives the distribution function, `qcusp` gives the quantile function, and `rcusp` generates observations.

Author(s)

Raoul Grasman

References

See cusp-package, integrate

See Also

cusp-package

Examples

```
# evaluate density and distribution
dcusp(0,2,3)
pcusp(0,2,3)
pcusp(qcusp(0.125,2,3),2,3) # = 0.125

# generate cusp variates
rcusp(100, 2, 3)

# generate cusp variates for random normal and splitting factor values
alpha = runif(20, -3, 3)
beta =  runif(20, -3, 3)
Vectorize(rcusp)(1, alpha, beta)
```

draw.cusp.bifset *Add Cusp Bifurcation Set Diagram to Existing Plot*

Description

Add a miniature bifurcation set for the cusp catastrophe to an existing plot.

Usage

```
draw.cusp.bifset(rx = par("usr")[1:2], ry = par("usr")[3:4], xpos = min(rx) +
    0.01 * diff(rx)[1], ypos = max(ry) - 0.01 * diff(ry)[1],
    xscale =  0.1 * diff(rx), yscale =  0.1 * diff(ry) / xscale,
    aspect = 1, mark = 1, col = hsv(0.7, s = 0.8, alpha = 0.5),
    border = NA, density = NA, bifurcation.set.fill = gray(0.8),
    background = hsv(0.1, s = 0.1, alpha = 0.5), ..., X)
```

Arguments

`rx`	x-axis range of the plot window
`ry`	y-axis range of the plot window
`xpos`	x-axis position of drawing
`ypos`	y-axis position of drawing
`xscale`	scaling applied to drawing along x-axis
`yscale`	scaling applied to drawing along y-axis
`aspect`	aspect ratio
`mark`	0, 1, 2, 3, or 4; indicates which part of the cusp surface should be marked
`col`	color used for marking a part of the cusp surface
`border`	color used for the marked part of the cusp surface. See `polygon` for details.
`density`	the density of shading lines of the marked part of the cusp surface, in lines per inch. The default value of NULL means that no shading lines are drawn. See `polygon` for details.
`bifurcation.set.fill`	color for marking the bifurcation set
`background`	background color of the cusp surface
`...`	arguments passed to `rect` and `polygon`
`X`	`data.frame`, deprecated

Details

This function is mainly intended for internal use by `cusp.plot`.

Value

No return value. Called for its side effect.

Author(s)

Raoul Grasman

References

http://www.scholarpedia.org/article/Cusp_bifurcation

See Also

plot.cusp, polygon

Examples

```
## Not run:
plot(1:10)
draw.cusp.bifset(mark=0) # no marking

## End(Not run)
```

---

oliva *Synthetic cusp data set*

Description

Synthetic ‘multivariate’ data from the cusp catastrophe as generated from the equations specified by Oliva et al. (1987).

Usage

```
data(oliva)
```

Format

A data frame with 50 observations on the following 12 variables.

x1 splitting factor predictor

x2 splitting factor predictor

x3 splitting factor predictor

y1 the bifurcation factor predictor

y2 the bifurcation factor predictor

y3 the bifurcation factor predictor

y4 the bifurcation factor predictor

z1 the state factor predictor

z2 the state factor predictor

alpha the true $alpha$'s

beta the true $beta$'s

y the true state variable values

Details

The data in Oliva et al. (1987) are obtained from the equations

$$\alpha_i = X_{i1} - .969\, X_{i2} - .201\, X_{i3},$$

$$\beta_i = .44\, Y_{i1} + 0.08\, Y_{i2} + .67\, Y_{i3} + .19\, Y_{i4},$$

$$y_i = -0.52\, Z_{i1} - 1.60\, Z_{i2}.$$

Here the X_{ij}'s are uniformly distributed on (-2,2), and the Y_{ij}'s and Z_{i1} are uniform on (-3,3). The states y_i were then generated from the cusp density, using rcusp, with their respective α_i's and β_i's as normal and splitting factors, and then Z_2 was computed as

$$Z_{i2} = (y_i + 0.52 Z_{i1})/(1.60).$$

Source

Oliva T, DeSarbo W, Day D, Jedidi K (1987). GEMCAT: A general multivariate methodology for estimating catastrophe models. Behavioral Science, 32(2), 121137.

References

Oliva T, Desarbo W, Day D, Jedidi K (1987). GEMCAT: A general multivariate methodology for estimating catastrophe models. Behavioral Science, 32(2), 121137.

Examples

```
data(oliva)
set.seed(121)
fit <- cusp(y ~ z1 + z2 - 1,
alpha ~ x1 + x2 + x3 - 1, ~ y1 + y2 + y3 + y4 - 1,
data = oliva, start = rnorm(9))
summary(fit)
## Not run:
cusp3d(fit, B=5.25, n.surf=50, theta=150)
# B modifies the range of beta (is set here to 5.25 to make
# sure all points lie on the surface)

## End(Not run)
```

---

plot.cusp *Graphical Diagnostic Display of Cusp Catastrophe Data Fit*

Description

This function generates diagnostic graphical displays of fits of a cusp catastrophe model to data obtained with cusp

Usage

```
## S3 method for class 'cusp'
plot(x, what = c("all", "bifurcation", "residual", "densities"), ...)
```

Arguments

`x`	Object returned by `cusp`
`what`	1-character string giving the type of plot desired. The following values are possible: `"all"` for a panel plot with all diagnostic plots, `"bifurcation"` for a plot of the bifurcation surface with estimated control parameter locations superimposed, "residual" for a plot of the residuals against fitted values, "densities" for a plot of density estimates conditioned on the estimated location on the bifurcation surface.
`...`	named arguments that are passed to lower level plotting function

Details

These diagnostic plots help to identify problems with the fitted model. In optimal cases the fitted locations in the parameter plane are dispersed over regions of qualitatively different behavior. Within each region the fitted dependent values have a density of the appropriate shape (e.g., bimodal in the bifurcation set).

Value

No return value. Called for its side effect.

Author(s)

Raoul Grasman

References

See `cusp-package`

See Also

`plotCuspBifurcation`, `plotCuspResidfitted`, `plotCuspDensities`

Examples

```
set.seed(20)
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)
## Not run:
plot(fit)

# just densities
```

```
layout(matrix(1:4,2))
plot(fit, what="densities")

## End(Not run)
```

---

plotCuspBifurcation *Display Fitted Data on Control Plane of Cusp Catastrophe.*

Description

Displays fitted data points on the control plane of cusp catastrophe. The function takes a fit object obtained with cusp and generates a plot. Different diagnostic plots may be chosen, or all can be combined in a single plot (the default).

Usage

```
plotCuspBifurcation(object, xlim = a + c(-0.3, 0.3), ylim = b + c(-0.1,
                    0.1), xlab = expression(alpha), ylab =
                    expression(beta), hue = 0.5 + 0.25 * tanh(object$y),
                    col = hsv(h = hue, s = 1, alpha = 0.4), cex.xlab =
                    1.55, cex.ylab = cex.xlab, axes = TRUE, box = TRUE,
                    add = FALSE, bifurcation.set.fill = gray(0.8),
                    cex.scale = 15, cex = (cex.scale/log(NROW(ab))) *
                    dens/max(dens), pch = 20)
```

Arguments

`object`	object returned by cusp
`xlim`	the x limits (x1, x2) of the plot.
`ylim`	the y limits of the plot.
`xlab`	a label for the x axis.
`ylab`	a label for the x axis.
`hue`	hue of points (see `hsv`)
`col`	color used in plots
`cex.xlab, cex.ylab`	see `par`
`axes`	logical. Should the axes be displayed?
`box`	logical. Should a box be drawn around the plot?
`add`	logical. Add to current plot?
`bifurcation.set.fill`	1-character string. Color used to fill the bifurcation set (see `colors`).
`cex.scale, cex, pch`	see `par`

Details

The default hue of each dot is a function of the height of the cusp surface to which it is closest. This is especially useful in the bifurcation set. Purple dots are higher than green dots.

The size of the dots depends on the density of dots at its location. The higher the density the larger the dot.

Value

No return value. Called for its side effect.

Author(s)

Raoul Grasman

References

See `cusp-package`

See Also

`plot.cusp`, `cusp3d`

Examples

```
set.seed(20)
# example with regressors
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)
## Not run:
plot(fit, what='bifurcation', box=TRUE, axes=FALSE)

## End(Not run)
```

`plotCuspDensities` *Plot Cusp State Variable Densities Conditioned on Control Parameter Values*

Description

Plot density of state variables conditioned on their location on the cusp control surface.

Usage

```
plotCuspDensities(object, main = "Conditional density", ...)
```

Arguments

`object`	cusp fit object returned by `cusp`
`main`	title of plot
`...`	named arguments that are passed to `plot` and `draw.cusp.bifset`

Details

This function is mainly intended for internal use by `plot.cusp`.

Value

No return value. Called for its side effect.

Author(s)

Raoul Grasman

See Also

`plot.cusp`

`plotCuspResidfitted` *Residuals against Fitted Plot for Cusp Model Fit*

Description

Plot Residuals against Fitted Values for a Cusp Model Fit.

Usage

```
plotCuspResidfitted(object, caption = "Residual vs Fitted",
   xlab = paste("Fitted (", colnames(fitted(object))[1], " convention)", sep = ""),
    ylab = "Residual", ...)
```

Arguments

`object`	cusp fit object returned by `cusp`
`caption`	plot caption
`xlab`	label for x-axis
`ylab`	label for y-axis
`...`	named arguments that are passed to `plot`

Details

This function is mainly intended for internal use by `plot.cusp`.

Value

No return value. Called for its side effect.

Author(s)

Raoul Grasman

See Also

plot.cusp

---

predict.cusp *Predict method for Cusp Model Fits*

Description

Predicted values based on a cusp model object.

Usage

```
## S3 method for class 'cusp'
predict(object, newdata, se.fit = FALSE, interval =
  c("none", "confidence", "prediction"), level = 0.95, type = c("response", "terms"),
   terms = NULL, na.action = na.pass, pred.var = res.var/weights, weights = 1,
  method = c("delay", "maxwell", "expected"), keep.linear.predictors = FALSE, ...)
```

Arguments

object	Object of class "cusp"
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
se.fit	See predict.lm. Not yet used.
interval	See predict.lm. Not yet used.
level	See predict.lm. Not yet used.
type	See predict.lm. Not yet used.
terms	See predict.lm. Not yet used.
na.action	See predict.lm. Not yet used.
pred.var	See predict.lm. Not yet used.
weights	See predict.lm. Not yet used.
method	Type of prediction convention to use. Can be abbreviated. (expected should currently not be trusted).
keep.linear.predictors	Logical. Should the linear predictors (alpha, beta, and y) be returned?
...	further arguments passed to or from other methods.

Details

predict.cusp produces predicted values, obtained by evaluating the regression functions from the cusp object in the frame newdata using predict.lm. This results in linear predictors for the cusp control variables alpha, and beta, and, if method = "delay", for the behavioral cusp variable y. These are then used to compute predicted values: If method = "delay" these are the points $y*$ on the cusp surface defined by

$$V'(y*) = \alpha + \beta y * - y*^3 = 0$$

that are closest to y. If method = "maxwell" they are the points on the cusp surface corresponding to the minimum of the associated potential function $V(y*) = \alpha y * + 0.5 y *^2 - 0.25 y*^4$.

Value

A vector of predictions. If keep.linear.predictors the return value has a "data" attribute which links to newdata augmented with the linear predictors alpha, beta, and, if method = "delay", y. If method = "expected", the expected value from the equilibrium distribution of the stochastic process

$$dY_t = V'(Y_t; \alpha, \beta)dt + dW_t,$$

where W_t is a Wiener proces (aka Brownian motion) is returned. (This distribution is implemented in dcusp.)

Note

Currently method = "expected" should not be trusted.

Author(s)

Raoul Grasman

References

See cusp-package.

See Also

cusp-package, predict.lm.

Examples

```
set.seed(123)
# example with regressors
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)

newdata = data.frame(x1 = runif(10), x2 = runif(10), z = 0)
predict(fit, newdata)
```

summary.cusp	*Summarizing Cusp Catastrophe Model Fits*

Description

summary method for class "cusp"

Usage

```
## S3 method for class 'cusp'
summary(object, correlation = FALSE, symbolic.cor = FALSE, logist = FALSE, ...)

## S3 method for class 'summary.cusp'
print(x, digits = max(3, getOption("digits") - 3), symbolic.cor = x$symbolic.cor,
    signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	Object returned by cusp
x	'summary.cusp' object
correlation	logical; if TRUE the correlation matrix is returned
symbolic.cor	logical; currently unused
logist	logical. If TRUE, a logistic model is fitted for cusp model assessment (see cusp.logist for details).
digits	numeric; the number of significant digits to use when printing.
signif.stars	logical. If TRUE, significance stars are printed for each coefficient.
...	further arguments passed to or from other methods.

Details

print.summary.cusp tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives significance stars if signif.stars is TRUE.

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print summary(object)$correlation directly.

Value

The function summary.cusp computes and returns a list of summary statistics of the fitted linear model given in object, using the components (list elements) "call" and "terms" from its argument, plus

call	the matched call
terms	the terms object used.
deviance	sum of squared residuals of cusp model fit

aic	Akaike Information Criterion for cusp model fit
contrasts	contrasts used
df.residual	degrees of freedom for the residuals of the cusp model fit
null.deviance	variance of canonical state variable
df.null	degrees of freedom of constant model for state variable
iter	number of optimization iterations
deviance.resid	residuals computed by `residuals.glm` using `type="deviance"`
coefficients	a $p \times 4$ matrix with columns for the estimated coefficient, its standard error, t-statistic and corresponding (two-sided) p-value. Aliased coefficients are omitted.
aliased	named logical vector showing if the original coefficients are aliased.
dispersion	always 1
df	3-vector containing the rank of the model matrix, residual degrees of freedom, and model degrees of freedom.
resid.name	string specifying the convention used in determining the residuals (i.e., "Delay" or "Maxwell").
cov.unscaled	the unscaled (dispersion = 1) estimated covariance matrix of the estimated coefficients.
r2lin.r.squared	R^2, the 'fraction of variance explained' by the linear regression model $$w_0 + w_1 Y_{i1} + \cdots + w_p Y_{ip} = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_q X_{iq} + \epsilon_i,$$ where Y contains all explanatory variables for the behavioral states in the cusp model, and X containes all explanatory variables for the control parameters of the cusp model. This is computed from the largest canonical correlation.
r2lin.dev	residual sums of squares of the linear model
r2lin.df	degrees of freedom for the linear model
r2lin.logLik	value of the log-likelihood for the linear model assuming normal errors
r2lin.npar	number of parameters in the linear model
r2lin.aic	AIC for the linear model
r2lin.aicc	corrected AIC for the linear model
r2lin.bic	BIC for the linear model
r2log.r.squared	R^2, the 'fraction of variance explained' by the logistic model. See `cusp.logist` for details.
r2log.dev	if `logist` = `TRUE` residual sums of square for the logistic model
r2log.df	ditto, degrees of freedom for the logistic model
r2log.logLik	ditto, value of log-likelihood function for the logistic model assuming normal errors.
r2log.npar	ditto, number of parameters for the logistic model

`r2log.aic`	ditto, AIC for logistic model
`r2log.aicc`	ditto, corrected AIC for logistic model
`r2log.bic`	ditto, BIC for logistic model
`r2cusp.r.squared`	pseudo-R^2, the 'fraction of variance explained by the cusp model', $$R^2 = 1 - \frac{Var(residuals_i)}{Var(y_i)}.$$ This value can be negative.
`r2cusp.dev`	residual sums of squares for cusp model
`r2cusp.df`	residual degrees of freedom for cusp model
`r2cusp.logLik`	value of the log-likelihood function for the cusp model
`r2cusp.npar`	number of parameters in the cusp model
`r2cusp.aic`	AIC for cusp model fit
`r2cusp.aicc`	corrected AIC for cusp model fit
`r2cusp.bic`	BIC for cusp model fit.

Author(s)

Raoul Grasman

References

Cobb L, Zacks S (1985). *Applications of Catastrophe Theory for Statistical Modeling in the Biosciences.* Journal of the American Statistical Association, 80(392), 793–802.

Hartelman PAI (1997). *Stochastic Catastrophe Theory.* Amsterdam: University of Amsterdam, PhD thesis.

Cobb L (1998). *An Introduction to Cusp Surface Analysis.*
https://www.aetheling.com/models/cusp/Intro.htm.

See Also

`cusp`, `cusp.logist`

Examples

```
set.seed(97)
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)
print(fit)
summary(fit, logist=FALSE) # set logist to TRUE to compare to logistic fit
```

`vcov.cusp` *Calculate Variance-Covariance Matrix for a Fitted Cusp Model Object*

Description

Returns an estimate of the variance-covariance matrix of the main parameters of a fitted cusp model object.

Usage

```
## S3 method for class 'cusp'
vcov(object, ...)
## S3 method for class 'cusp'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	a fitted cusp model object.
`parm`	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the confidence level required.
`...`	additional arguments for method functions.

Details

The variance-covariance matrix is estimated by the inverse of the Hessian matrix of the log-likelihood at the maximum likelihood estimate (`vcov`).

Normal theory confidence intervals are computed for all parameters in the cusp model object using `vcov` to obtain the standard errors (`confint`).

Value

The variance-covariance matrix (`vcov`).

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labeled as (1-level)/2 and 1 - (1-level)/2 in

Author(s)

Raoul Grasman

References

Seber, Wild (2005) Nonlinear regression. New York: Wiley

See Also

vcov, cusp

Examples

```
set.seed(123)
x1 = runif(150)
x2 = runif(150)
z = Vectorize(rcusp)(1, 4*x1-2, 4*x2-1)
data <- data.frame(x1, x2, z)
fit <- cusp(y ~ z, alpha ~ x1+x2, beta ~ x1+x2, data)

vcov(fit)
```

---

zeeman *Measurements from Zeeman's Catastrophe Machine*

Description

Data sets with measurements from different physical instances of Zeeman's Catastrophe Machine

Usage

```
data(zeeman1)
data(zeeman2)
data(zeeman3)
```

Format

A data frame with 150/198/282 observations on the following 3 variables.

x A control plane variable that is manipulable by the experimentalist.

y A control plane variable that is manipulable by the experimentalist.

z The state variable of the machine: the shortest distance to the longitudinal axis of the machine.

Details

The behavior Zeeman's catastrophe machine is archetypal for the Cusp catastrophe. This device consists of a wheel is tethered by an elastic chord to a fixed point. Another elastic, also attached to the wheel is moved about in the 'control plane' area opposite to the fixed point. The shortest distance between the strap point on the wheel and the axis defined by the fixed point and the control plane is recorded as a function of the position in the control plane. (In the original machine the angle between this axis and the line through the wheel center and the strap point is used.) See https://www.math.stonybrook.edu/~tony/whatsnew/column/catastrophe-0600/cusp4.html for a vivid demonstration. These data sets were obtained from 3 different physical instances of this machine, made by different people.

Measurements were made by systematically sampling different points in the control plane.

See vignette for example analysis with all three data sets.

For pictures of the machines, see

Zeeman catastrophy machine 1 https://purl.oclc.org/net/rgrasman/cusp/zeeman1

Zeeman catastrophy machine 2 https://purl.oclc.org/net/rgrasman/cusp/zeeman2

Zeeman catastrophy machine 3 https://purl.oclc.org/net/rgrasman/cusp/zeeman3

Source

`zeeman1` is due to Noemi Schuurman

`zeeman2` is due to Karin Visser

`zeeman3` is due to Mats Nagel & Joris ?

See https://sites.google.com/site/zeemanmachine/data-repository

References

Zeeman (1976).

Examples

```
data(zeeman1)
data(zeeman2)
data(zeeman3)
## Not run:
fit <- cusp(y~z, alpha~x+y, beta~x+y, data=zeeman1)
plot(fit)
cusp3d(fit, surf.hue = 40, theta=215, phi=37.5, B=5.25)

## End(Not run)
```

Index