Package 'RSA'

July 2, 2014

Encoding UTF-8

Type Package

Title Response surface analysis

Version 0.9.3

Date 2014-06-27

Author Felix Schönbrodt

Maintainer Felix Schönbrodt <felix@nicebread.de>

Description Advanced response surface analysis. The main function RSA computes and compares several nested polynomial regression models (full polynomial,shifted and rotated squared differences, rising ridge surfaces, basic squared differences). The package provides plotting functions for 3d wireframe surfaces, interactive 3d plots, and contour plots. Calculates many surface parameters (a1 to a4, principal axes, stationary point,eigenvalues) and provides standard, robust, or bootstrapped standard errors and confidence intervals for them.

Suggests fields, SDMTools, tkrplot, rgl, qgraph, aplpack, AICcmodavg

Depends R (>= 2.15.0), lavaan (>= 0.5.11), ggplot2, lattice, gridExtra

Imports plyr, RColorBrewer

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-27 09:03:35

R topics documented:

`aictab` *Show a table of AIC model comparisons*

Description

Show a table of AIC model comparisons

Usage

```
aictab(x, plot = FALSE)
```

Arguments

`x`	An RSA object
`plot`	Should a plot of the AICc table be plotted? (Experimental)

Details

TODO

compare *Compare a full list of RSA models*

Description

Compare several fit indexes of all models computed from the RSA function

Usage

```
compare(x, verbose = TRUE, plot = FALSE, ...)
```

Arguments

x	An RSA object
verbose	Should the summary be printed?
plot	Should the comparison be plotted (using the modeltree function)?
...	Additional parameters passed to the modeltree function

Details

No details so far.

compare2 *Compare two specific RSA models*

Description

Compare several fit indexes of two models computed from the RSA function

Usage

```
compare2(x, m1 = "", m2 = "full", verbose = TRUE)
```

Arguments

x	An RSA object
m1	Name of first model
m2	Name of second model
verbose	Should the summary be printed?

Details

You must take care yourself that the compared models are nested! There is no automatic check, so you could, in principle, compare non-nested models. This is valid for AIC, BIC, CFI, TLI, and R2 indices, but *not* for the chi2-LR test!

confint.RSA *Computes confidence intervals for RSA parameters, standard or bootstrapped*

Description

Computes confidence intervals for RSA parameters, standard or bootstrapped

Usage

```
## S3 method for class 'RSA'
confint(object, parm, level = 0.95, ..., model = "full",
  digits = 3, method = "standard", R = 5000)
```

Arguments

object	An RSA object
parm	Not used.
level	The confidence level required.
digits	Number of digits the output is rounded to; if NA, digits are unconstrained
model	A string specifying the model; defaults to "full"
method	"standard" returns the CI for the lavaan object as it was computed. "boot" computes new percentile bootstrapped CIs.
R	If `method = "boot"`, R specifies the number of bootstrap samples
...	Additional parameters passed to the bootstrapLavaan function, e.g., `parallel="multicore", ncpus=2`.

Details

None so far.

See Also

RSA

Examples

```
set.seed(0xBEEF)
n <- 300
err <- 2
x <- rnorm(n, 0, 5)
y <- rnorm(n, 0, 5)
df <- data.frame(x, y)
df <- within(df, {
diff <- x-y
absdiff <- abs(x-y)
SD <- (x-y)^2
```

```
z.sq <- SD + rnorm(n, 0, err)
})

r1 <- RSA(z.sq~x*y, df, models="SSQD")
(c1 <- confint(r1, model="SSQD"))

# Dummy example with 10 bootstrap replications - better use >= 5000!
(c2 <- confint(r1, model="SSQD", method="boot", R=10))
## Not run:
# multicore version
confint(r1, model="SSQD", R=5000, parallel="multicore", ncpus=2)

## End(Not run)
```

---

demoRSA *Plots a response surface of a polynomial equation of second degree with interactive controls*

Description

Plots an RSA object, or a response surface with specified parameters, with interactive controls for coefficients.

Usage

```
demoRSA(x = NULL, y = 0, x2 = 0, y2 = 0, xy = 0, w = 0, wx = 0,
  wy = 0, x3 = 0, xy2 = 0, x2y = 0, y3 = 0, b0 = 0, type = "3d",
  zlim = c(-2, 2), xlim = c(-2, 2), ylim = c(-2, 2), points = list(show
  = TRUE, value = "raw", jitter = 0, color = "black", cex = 0.5),
  model = "full", extended = FALSE, ...)
```

Arguments

x	Either an RSA object (returned by the RSA function), or the coefficient for the X predictor
y	Y coefficient
x2	X^2 coefficient
y2	Y^2 coefficient
xy	XY interaction coefficient
y3	Y^3 coefficient
x3	X^3 coefficient
xy2	XY^2 coefficient
x2y	X^2Y coefficient
w	W coefficient (for (un)constrained absolute difference model)
wx	WX coefficient (for (un)constrained absolute difference model)

wy	WY coefficient (for (un)constrained absolute difference model)
b0	Intercept
xlim	Limits of the x axis
ylim	Limits of the y axis
zlim	Limits of the z axis
type	3d for 3d surface plot, contour for 2d contour plot. Shortcuts (i.e., first letter of string) are sufficient; be careful: "contour" is very slow at the moment
points	A list of parameters which define the appearance of the raw scatter points: show = TRUE: Should the original data points be overplotted? value="raw": Plot the original z value, "predicted": plot the predicted z value. jitter=0: Amount of jitter for the raw data points. cex = .5: multiplication factor for point size.
model	If x is an RSA object: from which model should the response surface be computed?
extended	Show additional controls (not implemented yet)
...	Other parameters passed through to plot.RSA (e.g., xlab, ylab, zlab, cex, legend)

Details

No details so far. Just play around with the interface!

See Also

plotRSA, RSA

Examples

```
# Plot response surfaces from known parameters
# example of Edwards (2002), Figure 3
## Not run:
demoRSA(x=.314, y=-.118, x2=-.145, y2=-.102, xy=.299, b0=5.628, type="3d")
demoRSA(x=.314, y=-.118, x2=-.145, y2=-.102, xy=.299, b0=5.628, legend=FALSE, type="c")

## End(Not run)

# Plot response surface from an RSA object
## Not run:
set.seed(0xBEEF)
n <- 300
err <- 2
x <- rnorm(n, 0, 5)
y <- rnorm(n, 0, 5)
df <- data.frame(x, y)
df <- within(df, {
diff <- x-y
absdiff <- abs(x-y)
SD <- (x-y)^2
z.diff <- diff + rnorm(n, 0, err)
z.abs <- absdiff + rnorm(n, 0, err)
```

```
z.sq <- SD + rnorm(n, 0, err)
z.add <- diff + 0.4*x + rnorm(n, 0, err)
z.complex <- 0.4*x + - 0.2*x*y + + 0.1*x^2 - 0.03*y^2 + rnorm(n, 0, err)
})

r1 <- RSA(z.sq~x*y, df)
demoRSA(r1)
demoRSA(r1, points=TRUE, model="SQD")

## End(Not run)
```

`fitted.RSA` *Return fitted values of a RSA model*

Description

Return fitted values of a RSA model

Usage

```
## S3 method for class 'RSA'
fitted(object, ..., model = "full")
```

Arguments

`object`	An RSA object.
`...`	Other parameters (currently not used)
`model`	Model on which the fitted values are based

`getPar` *Retrieves several variables from an RSA object*

Description

Retrieves several variables from an RSA object

Usage

```
getPar(x, type = "coef", model = "full", digits = NA, ...)
```

Arguments

`x`	RSA object
`type`	One of: "syntax", "coef", "R2", "R2.adj", "free", "summary"
`model`	A string specifying the model; defaults to "full"
`digits`	Number of digits the output is rounded to; if NA, digits are unconstrained
`...`	Additional parameters passed to the extraction function

Details

None so far.

See Also

RSA

Examples

```
set.seed(0xBEEF)
n <- 300
err <- 2
x <- rnorm(n, 0, 5)
y <- rnorm(n, 0, 5)
df <- data.frame(x, y)
df <- within(df, {
diff <- x-y
absdiff <- abs(x-y)
SD <- (x-y)^2
z.sq <- SD + rnorm(n, 0, err)
})

r1 <- RSA(z.sq~x*y, df, models=c("full", "SSQD"))
getPar(r1, "syntax")
getPar(r1, "R2")
getPar(r1, "coef")
```

---

modeltree *Plots a flow chart with model comparisons*

Description

Plots a flow chart with model comparisons from a RSA object

Usage

```
modeltree(x, digits = 3, sig = 0.05, borderline = 0.1, ...)
```

Arguments

x	A cRSA object (= output from the compare function)
digits	The number of digits to which numbers are rounded
sig	Threshold for models to be marked as "not significant"
borderline	Threshold for models to be marked as "borderline significant" (used for color of arrows)
...	Additional parameters (not used yet)

Details

The plot can be either requested within the `compare` function: `compare(r1, plot=TRUE)` Or it can be plotted from a cRSA object (= output from the `compare` function): `c1 <- compare(r1)` `plot(c1)`

See Also

`RSA`, `compare`

---

motcon *Data set on motive congruence.*

Description

A dataset containing the explicit power motive, implicit power motive and self ratings of affective valence during a spontaneous speech. The variables are as follows:

Format

A data frame with 84 rows and 3 variables

Details

- ePow Explicit power motive, measured with a questionnaire (Unified Motive Scales, Schönbrodt & Gerstenberg, 2012). Raw values have been z standardized.
- iPow Implicit power motive, measure with picture story exercise (6 pictures). Raw motive scores have been controlled for word count and z standardized
- postVA z standardized valence rating after the speech ('How did you feel during the speech'). Consists of two bipolar items from the PANAVA questionnaire (Schallberger, 2005): 'zufrieden ... unzufrieden' (satisfied ... unsatisfied) and 'unglücklich ... glücklich' (unhappy ... happy).

References

Schallberger, U. (2005). *Kurzskala zur Erfassung der Positiven Aktivierung, Negativen Aktivierung und Valenz in Experience Sampling Studien (PANAVA-KS) [Short scales for the assessment of positive affect, negative affect, and valence in experience sampling studies]*. University of Zurich.

Schönbrodt, F. D., & Gerstenberg, F. X. R. (2012). An IRT analysis of motive questionnaires: The Unified Motive Scales. *Journal of Research in Personality, 46*, 725-742. doi:10.1016/j.jrp.2012.08.010

movieRSA *Create a movie of plotRSA plots, with changing surface and/or rotation*

Description

Create a movie of plotRSA plots, with changing surface and/or rotation

Usage

```
movieRSA(name, frames, dur = 2000, fps = 30, width = 800, height = 600,
  mirror = TRUE, savetodisk = TRUE, clean = TRUE)
```

Arguments

name	Name for the subfolder containing all still pictures, and for the final movie file.
frames	A list of lists: Each list contains parameters which are passed to the plotRSA function. See plotRSA for details.
dur	Duration of the movie in miliseconds
fps	Frame per second (defaults to 30)
width	Width of the final movie in pixels
height	Height of the final movie in pixels
mirror	If TRUE, the frame sequence is mirrored at the end so that the movie ends at frame 1.
savetodisk	If TRUE the files are saved to the disk. If FALSE, the movie is only shown on the screen
clean	Should the still images be deleted?

Details

frames is a list of the first, intermediate, and the final parameters of the surface. Each scalar parameter defined in frames is interpolated between steps in order to create a smooth sequence of plots. Logical and character parameters are inherited from the first frame. Plots are saved as individual still pictures in a subfolder called name and finally glued together using ffmpeg. Hence, a ffmpeg installation is needed to create the movie (the still pictures can be produced without ffmpeg).

See Also

plotRSA

Examples

```
## Not run:
movieRSA(name="SD0",
frames <- list(
  step1 = list(b0=0, xy=-.40, x2=.20, y2=.20,
rotation=list(x=-63, y=32, z=15),
legend=FALSE, zlim=c(0, 4), param=FALSE),
  step2 = list(b0=0, xy=-.10, x2=.05, y2=.05,
rotation=list(x=-54, y=39, z=25)),
step3 = list(b0=0, xy=-.40, x2=.20, y2=.20,
rotation=list(x=-45, y=45, z=35))
 ),
 mirror=TRUE, fps=30, dur=5000)

## End(Not run)
```

---

plotRSA *Plots a response surface of a polynomial equation of second degree*

Description

Plots an RSA object, or a response surface with specified parameters

Usage

```
plotRSA(x = 0, y = 0, x2 = 0, y2 = 0, xy = 0, w = 0, wx = 0,
  wy = 0, x3 = 0, xy2 = 0, x2y = 0, y3 = 0, b0 = 0, type = "3d",
  model = "full", xlim = NULL, ylim = NULL, zlim = NULL, xlab = NULL,
  ylab = NULL, zlab = NULL, main = "", surface = "predict",
  lambda = NULL, rotation = list(x = -63, y = 32, z = 15),
  label.rotation = list(x = 19, y = -40, z = 92), gridsize = 21,
  bw = FALSE, legend = TRUE, param = TRUE, axes = c("LOC", "LOIC",
  "PA1", "PA2"), project = FALSE, maxlines = FALSE, cex = 1,
  cex.main = 1, points = list(data = NULL, show = NA, value = "raw", jitter
  = 0, color = "black", cex = 0.5, out.mark = FALSE), demo = FALSE,
  fit = NULL, link = "identity", tck = c(1.5, 1.5, 1.5),
  distance = c(1.3, 1.3, 1.4), border = TRUE, contour = list(show = FALSE,
  color = "grey40", highlight = c()), hull = NA, SP.CI = FALSE,
  pal = NULL, pal.range = "box", pad = 0, ...)
```

Arguments

x	Either an RSA object (returned by the RSA function), or the coefficient for the X predictor
y	Y coefficient
x2	X^2 coefficient
y2	Y^2 coefficient

xy	XY interaction coefficient
w	W coefficient (for (un)constrained absolute difference model)
wx	WX coefficient (for (un)constrained absolute difference model)
wy	WY coefficient (for (un)constrained absolute difference model)
y3	Y^3 coefficient
x3	X^3 coefficient
xy2	XY^2 coefficient
x2y	X^2Y coefficient
b0	Intercept
xlim	Limits of the x axis
ylim	Limits of the y axis
zlim	Limits of the z axis
xlab	Label for x axis
ylab	Label for y axis
zlab	Label for z axis
main	the main title of the plot
cex.main	Factor for main title size
surface	Method for the calculation of the surface z values. "predict" takes the predicted values from the model, "smooth" uses a thin plate smoother (function Tps from the fields package) of the raw data
lambda	lambda parameter for the smoother. Default (NULL) means that it is estimated by the smoother function. Small lambdas around 1 lead to rugged surfaces, big lambdas to very smooth surfaces.
rotation	Rotation of the 3d surface plot (when type == "3d")
label.rotation	Rotation of the axis labls (when type == "3d")
gridsize	Number of grid nodes in each dimension
bw	Print surface in black and white instead of colors?
legend	Print color legend for z values?
cex	Font size factor for axes labels, axes titles, and the main title
type	3d for 3d surface plot, contour for 2d contour plot, "interactive" for interactive rotatable plot. Shortcuts (i.e., first letter of string) are sufficient
points	A list of parameters which define the appearance of the raw scatter points: • data: Data frame which contains the coordinates of the raw data points. First column = x, second = y, third = z. This data frame is automatically generated when the plot is based on a fitted RSA-object • show = TRUE: Should the original data points be overplotted? • color = "black": Color of the points • value="raw": Plot the original z value, "predicted": plot the predicted z value

- jitter = 0: Amount of jitter for the raw data points. For z values, a value of 0.005 is reasonable
- cex = .5: multiplication factor for point size
- out.mark = FALSE: If set to TRUE, outliers according to Bollen & Jackman (1980) are printed as red X symbols, but only when they have been removed in the RSA function: `RSA(..., out.rm=TRUE)`.
  - If out.rm == TRUE (in RSA()) and out.mark == FALSE (in plotRSA()), the outlier is removed from the model and *not plotted* in plotRSA.
  - If out.rm == TRUE (in RSA()) and out.mark == TRUE (in plotRSA()), the outlier is removed from the model but plotted and marked in plotRSA.
  - If out.rm == FALSE (in RSA()): Outliers are not removed and cannot be plotted.
  - Example syntax: `plotRSA(r1, points=list(show=TRUE, out.mark=TRUE))`

`model` If x is an RSA object: from which model should the response surface be computed?

`demo` Do not change that parameter (internal use only)

`fit` Do not change that parameter (internal use only)

`param` Should the surface parameters a1 to a4 be shown on the plot? In case of a 3d plot a1 to a4 are printed on the upper left side; in case of a contour plot the principal axes are plotted.

`axes` A vector of strings specifying the axes that should be plotted. Can be any combination of c("LOC", "LOIC", "PA1", "PA2"). LOC = line of congruence, LOIC = line of incongruence, PA1 = first principal axis, PA2 = second principal axis

`project` Should the LOC, LOIC, etc. (as defined in parameter `axes`) be also plotted as a projection on the bottom of the cube?

`maxlines` Should the maximum lines be plotted? (red: maximum X for a given Y, blue: maximum Y for a given X). Works only in type="3d"

`link` Link function to transform the z axes. Implemented are "identity" (no transformation; default), "probit", and "logit"

`border` Should a thicker border around the surface be plotted? Sometimes this border leaves the surrounding box, which does not look good. In this case the border can be suppressed by setting `border=FALSE`.

`contour` A list defining the appearance of contour lines (aka. height lines). show=TRUE: Should the contour lines be plotted on the 3d wireframe plot? (Parameter only relevant for `type="3d"`). color = "grey40": Color of the contour lines. highlight = c(): A vector of heights which should be highlighted (i.e., printed in bold). Be careful: the highlighted line is not necessarily exactly at the specified height; instead the nearest height line is selected.

`hull` Plot a bag plot on the surface (This is a bivariate extension of the boxplot. 50% of points are in the inner bag, 50% in the outer region). See Rousseeuw, Ruts, & Tukey (1999).

`SP.CI` Plot the CI of the stationary point (only relevant for `type="contour"`)

`distance` A vector of three values defining the distance of labels to the axes

tck	A vector of three values defining the position of labels to the axes (see ?wireframe)
pal	A palette for shading. You can use colorRampPalette to construct a color ramp, e.g. plot(r.m, pal=colorRampPalette(c("darkgreen", "yellow", "darkred"))(20))
pal.range	Should the color range be scaled to the box (pal.range = "box", default), or to the min and max of the surface (pal.range = "surface")? If set to "box", different surface plots can be compared along their color, as long as the zlim is the same for both.
pad	Pad controls the margin around the figure (positive numbers: larger margin, negative numbers: smaller margin) #'
...	Additional parameters passed to the plotting function (e.g., sub="Title"). A useful title might be the R squared of the plotted model: sub = as.expression(bquote(R^2==.(round(get

Details

Each plot type has its distinctive advantages. The two-dimensional contour plot gives a clear view of the position of the principal axes and the stationary point. The 3d plot gives a three dimensional impression of the surface, allows overplotting of the original data points (in case an RSA object is provided), and allows the interactive adjustment of regression weights in the demoRSA function. The interactive plot allows rotating and exploring a three-dimensional surface with the mouse (nice for demonstration purposes). If you want to export publication-ready plots, it is recommended to export it with following commands: p1 <- plot(r1, bw=TRUE) trellis.device(device="cairo_pdf", filename="RSA_plot.pdf") print(p1) dev.off()

References

Rousseeuw, P. J., Ruts, I., & Tukey, J. W. (1999). The Bagplot: A Bivariate Boxplot. The American Statistician, 53(4), 382-387. doi:10.1080/00031305.1999.10474494

See Also

demoRSA, RSA

Examples

```
# Plot response surfaces from known parameters
# example of Edwards (2002), Figure 3
# Default: 3d plot:
plotRSA(x=.314, y=-.118, x2=-.145, y2=-.102, xy=.299, b0=5.628)
# Contour plot:
plotRSA(x=.314, y=-.118, x2=-.145, y2=-.102, xy=.299, b0=5.628, type="c")
# Interactive plot (try the mouse!):
plotRSA(x=.314, y=-.118, x2=-.145, y2=-.102, xy=.299, b0=5.628, type="i")

# Plot response surface from an RSA object
set.seed(0xBEEF)
n <- 300
err <- 2
x <- rnorm(n, 0, 5)
```

```
y <- rnorm(n, 0, 5)
df <- data.frame(x, y)
df <- within(df, {
diff <- x-y
absdiff <- abs(x-y)
SD <- (x-y)^2
z.diff <- diff + rnorm(n, 0, err)
z.abs <- absdiff + rnorm(n, 0, err)
z.sq <- SD + rnorm(n, 0, err)
z.add <- diff + 0.4*x + rnorm(n, 0, err)
z.complex <- 0.4*x + - 0.2*x*y + + 0.1*x^2 - 0.03*y^2 + rnorm(n, 0, err)
})

r1 <- RSA(z.sq~x*y, df, models=c("SQD", "full", "IA"))
plot(r1) # default: model = "full"
plot(r1, model="SQD", points=list(show=TRUE, value="predicted"))
```

`residuals.RSA` *Return residual values of a RSA model*

Description

Return residual values of a RSA model

Usage

```
## S3 method for class 'RSA'
residuals(object, ..., model = "full")
```

Arguments

`object`	An RSA object.
`...`	Other parameters (currently not used)
`model`	Model on which the fitted values are based

`RSA` *Performs several RSA model tests on a data set with two predictors*

Description

Performs several RSA model tests on a data set with two predictors

Usage

```
RSA(formula, data = NULL, center = FALSE, scale = FALSE, na.rm = FALSE,
  out.rm = TRUE, breakline = FALSE, models = "default", cubic = FALSE,
  verbose = TRUE, add = "", control.variables = c(), ...)
```

Arguments

formula	A formula in the form `z ~ x*y`, specifying the variable names used from the data frame, where z is the name of the response variable, and x and y are the names of the predictor variables.
data	A data frame with the variables
center	Should predictor variables be centered on *each variable's* sample mean before analyses? You should think carefully about this option, as different centering of the predictor variables can affect the commensurability of the predictor scales.
scale	Should predictor variables be scales on the SD of *each variable* before analyses? You should think carefully about this option, as different scaling of the predictor variables can affect the commensurability of the predictor scales.
na.rm	Remove missings before proceeding?
add	Additional syntax that is added to the lavaan model. Can contain, for example, additional constraints, like "p01 == 0; p11 == 0"
out.rm	Should outliers according to Bollen & Jackman (1980) criteria be excluded from analyses? In large data sets this analysis is the speed bottleneck. If you are sure that no outliers exist, set this option to FALSE for speed improvements.
breakline	Should the breakline in the unconstrained absolute difference model be allowed (the breakline is possible from the model formulation, but empirically rather unrealistic ...). Defaults to `FALSE`
verbose	Should additional information during the computation process be printed?
models	A vector with names of all models that should be computed. Should be any from `c("absdiff", "absunc", "diff", "mean", "additive", "IA", "SQD", "RR", "SRR", "SRRR", "` For `models="all"`, all models are computed, for `models="default"` all models besides absolute difference models are computed.
cubic	Should a cubic model with the additional terms Y^3, XY^2, YX^2, and X^3 be included?
control.variables	A string vector with variable names from `data`. These variables are added as linear predictors to the model (in order "to control for them"). No interactions with the other variables are modeled.
...	Additional parameters passed to the `lavaan` `sem` function. For example, you can obtained bootstrapped standard errors by setting `se="boot"`.

Details

You can also fit binary outcome variables with a probit link function. For that purpose, the response variable has to be defined as "ordered": `r1 <- RSA(Z.binary ~ X*Y, dat, ordered="Z.binary")` (for more details see the help file of the `sem` function in the `lavaan` package.). The results can also be plotted with probabilities on the z axis using the probit link function: `plot(r1, link="probit", zlim=c(0, 1), zlab="Pr` `lavaan` at the moment only supports a probit link function for binary outcomes, not a logit link.

See Also

`demoRSA`, `plotRSA`, `RSA.ST`, `confint.RSA`

Examples

```
# Compute response surface from a fake data set
set.seed(0xBEEF)
n <- 300
err <- 15
x <- rnorm(n, 0, 5)
y <- rnorm(n, 0, 5)
df <- data.frame(x, y)
df <- within(df, {
diff <- x-y
absdiff <- abs(x-y)
SD <- (x-y)^2
z.diff <- diff + rnorm(n, 0, err)
z.abs <- absdiff + rnorm(n, 0, err)
z.sq <- SD + rnorm(n, 0, err)
z.add <- diff + 0.4*x + rnorm(n, 0, err)
z.complex <- 0.4*x + - 0.2*x*y + + 0.1*x^2 - 0.03*y^2 + rnorm(n, 0, err)
})
## Not run:
r1 <- RSA(z.sq~x*y, df)
print(r1)
compare(r1)
plot(r1)
plot(r1, model="SRSQD")
plot(r1, model="full", type="c")
getPar(r1, "coef") # print model parameters including SE and CI
RSA.ST(r1) # get surface parameters

# Motive congruency example
data(motcon)
r.m <- RSA(postVA~ePow*iPow, motcon)

# Get boostrapped CIs with 10 bootstrap samples (usually this should be set to 5000 or higher),
# only from the SSQD model
c1 <- confint(r.m, model="SSQD", method="boot", R=10)

# Plot the final model
plot(r.m, model="RR", xlab="Explicit power motive",
ylab="Implicit power motive", zlab="Affective valence")

## End(Not run)
```

RSA.ST *Surface tests*

Description

Calculates surface parameters a1 to a4, the stationary point, the principal axes, the eigenvectors and -values

Usage

```
RSA.ST(x, y = 0, x2 = 0, xy = 0, y2 = 0, b0 = 0, SE = NULL,
  COV = NULL, df = NULL, model = "full")
```

Arguments

x	Either an RSA object (returned by the RSA function), or the coefficient for the X predictor
y	Y coefficient
x2	X^2 coefficient
y2	Y^2 coefficient
xy	XY interaction coefficient
b0	The intercept
SE	In case that the coefficients are provided directly (as parameters x, y, x2, y2, xy), SE can provide the standard errors of these estimates. SE has to be a named vector, e.g.: SE=c(x=.1, y=.2, x2=.1, y2=.5, xy=.3). SEs of all parameters have to provided, otherwise the function will print an error. In case standard errors *and* the covariances (see below) *and* df (see below) are provided, parametric confidence intervals for a1 to a4 are calculated.
COV	Covariances between parameters. COV has to be a named vector, e.g.: COV=c(x_y=.1, x2_y2 = .2, x2_ where x_y is the covariance between x and y, and so on. All these covariances have to provided with exactly these names, otherwise the function will print an error.
df	Degrees of freedom for the calculation of a1 to a4 confidence intervals. The df are the residual dfs of the model (df = n - estimated parameters). For the full polynomial model, this is n - 6 (folllowing parameters are estimated: Intercept, x, y, xy, x2, y2).
model	If x is an RSA object, this parameter specifies the model from which to extract the coefficients

Details

No details so far.

Value

Returns surface parameters a1 to a4. If an RSA object or SE, COV and df are provided, also significance test and standard errors of a1 to a4 are reported. The stationary point (X0, Y0, and Z0). First principal axis (PA) relative to the X-Y plane (p10 = intercept, p11 = slope), second PA (p20 = intercept, p21 = slope). M = eigenvectors, l = eigenvalues, L = lambda matrix as1X to as4X: surface parameters of the PA, relative to X values as1Y to as4Y: surface parameters of the PA, relative to Y values PA1.curvature: quadratic component of the first PA (equivalent to the first eigenvalue) PA2.curvature: quadratic component of the second PA (equivalent to the second eigenvalue)

References

Shanock, L. R., Baran, B. E., Gentry, W. A., Pattison, S. C., & Heggestad, E. D. (2010). Polynomial Regression with Response Surface Analysis: A Powerful Approach for Examining Moderation and Overcoming Limitations of Difference Scores. *Journal of Business and Psychology, 25*, 543-554. doi:10.1007/s10869-010-9183-4

See Also

RSA

Examples

```
# get surface parameters from known parameters
# example from Shanock et al. (2010), p. 548, Table 2
RSA.ST(x=-.23, y=.77, x2=-.07, y2=-.10, xy=.27)

# Get surface parameters from a computed RSA object
set.seed(0xBEEF)
n <- 300
err <- 2
x <- rnorm(n, 0, 5)
y <- rnorm(n, 0, 5)
df <- data.frame(x, y)
df <- within(df, {
diff <- x-y
absdiff <- abs(x-y)
SD <- (x-y)^2
z.diff <- diff + rnorm(n, 0, err)
z.abs <- absdiff + rnorm(n, 0, err)
z.sq <- SD + rnorm(n, 0, err)
z.add <- diff + 0.4*x + rnorm(n, 0, err)
z.complex <- 0.4*x + - 0.2*x*y + + 0.1*x^2 - 0.03*y^2 + rnorm(n, 0, err)
})

r1 <- RSA(z.sq~x*y, df, models="full")
RSA.ST(r1)
```

Index